NAME | intro – introduces the PEXlib graphics library

DESCRIPTION | PEXlib is a programmer’s interface to the PEX protocol. PEX is an extension to the X window system for supporting three-dimensional graphics. The application can query the existence of the PEX server extension by calling `XQueryExtension` using the name, `PEX_NAME_STRING`, defined in PEXlib include files.

To begin using PEXlib, the application must initialize PEXlib by calling `PEXInitialize(3)` before calling any PEXlib function which takes a display argument. Functions which do not take a display argument may be called prior to calling `PEXInitialize(3)`. `PEXInitialize(3)` must be called for each display connection the application will use. Failure to call `PEXInitialize(3)` will result in undefined behavior for functions which take a display argument. Termination of PEXlib occurs implicitly at either program end or `XCloseDisplay`.

There are four other functions that should be called during initialization to determine what features and parameters are supported by each particular PEX server implementation. These functions are `PEXGetExtensionInfo(3)`, `PEXGetImpDepConstants(3)`, `PEXGetEnumTypeInfo(3)`, and `PEXMatchRenderingTargets(3)`. `PEXGetExtensionInfo(3)` returns information about the PEX server extension such as protocol version, vendor string, release number and subset support. `PEXGetImpDepConstants(3)` returns information about supported implementation-dependent constants. `PEXGetEnumTypeInfo(3)` returns information about supported enumerated types. `PEXMatchRenderingTargets(3)` returns information about supported drawable types.

Some PEXlib functions allocate memory on behalf of the application. Memory allocation often occurs in the case of inquiry functions where the size of data to be returned is unknown. Applications must use `XFree` to deallocate memory that has been allocated for them by PEXlib, unless a specific function is provided to free memory allocated for particular PEXlib objects. Where necessary, the specific function for deallocation is noted in the description of the function which allocates the memory. Memory allocated by PEXlib is considered to be "owned" by PEXlib; the application must not modify pointers to memory returned by the library nor attempt to free such memory except by the specified interfaces. Data structure fields other than pointers to memory can be changed without consequence.

PEXlib is designed as an extension to Xlib using the same transport and error mechanisms as Xlib. This includes all requests, replies, events and errors. This means synchronization (e.g. `XSync` and `XSynchronize`) will have the same effect on PEXlib requests as they do on Xlib requests. PEX events also are treated in the same way as X events (e.g. use `XNextEvent`).

PEXlib functions usually do not check for invalid parameters. Applications should be careful to pass correct data to PEXlib. Incorrect values sent to the PEX server extension will cause an error event to be returned to the client. These errors are asynchronous, and so are difficult to trace to a particular Xlib or PEXlib call unless using `XSync` or `XSynchronize`. Note these functions should be used with prudence, since they slow client programs significantly.

modified November 1995
By default, when an error event is returned by the server, Xlib prints a message and the application does not continue. The application can register its own error handler by calling `XSetErrorHandler`.

The floating point format used by PEXlib is the format native to the architecture of the client machine (i.e. the machine on which the application executes). All floating point values are expected to be in that native format, with a few exceptions for functions which allow the application to specify the floating point format. Those functions which allow the application to specify a floating point format deal primarily with application data that is already in protocol format, or data that the application wants to be encoded into protocol format for its own use.

When the server to which PEXlib is sending data does not support the floating point format native to PEXlib, PEXlib will choose an appropriate format which is supported by the server and convert into that format when formatting protocol to send to that server. In cases where PEXlib must choose a non-native floating point format, the application can determine the format chosen by calling This is useful with those functions dealing with protocol formatted data to help avoid unnecessary format conversions when sending the data to the PEX server.

The PEXlib specification currently defines function bindings and include files for the C language only. The specification is intended to work for both ANSI C and Kernighan and Ritchie (K&R) C compilers.

The `<X11/PEX5/PEX.h>` and `<X11/PEX5/PEXlib.h>` include files contain constant definitions, macros, and data structures used by PEXlib. `<X11/PEX5/PEXlib.h>` automatically includes `<X11/PEX5/PEX.h>`. Some data structures have fields named "reserved". These fields are necessary to align the data correctly for efficient transfer to the protocol transport buffer. Any values assigned to these fields will be ignored.

The library is specified in the compilation command as "-lPEX5".
NAME PEXAccumulateState – Accumulate Rendering Pipeline State

SYNTAX void PEXAccumulateState(Display *display, PEXRenderer renderer, unsigned long count, PEXElementRef *elements)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
renderer The resource identifier of a renderer resource.
count The number of elements.
elements A pointer to the structure element reference path.

RETURNS None

DESCRIPTION This function accumulates the state that would be in effect if a traversal were done to the element specified in the path. If the renderer state is PEXIdle, the request is ignored.

The accumulation of rendering pipeline state begins with the current pipeline attributes. A linear traversal down the specified path is then made and the structure elements that lie along the specified path are examined in order. Any element that contains an output command that would modify the pipeline state (i.e. output attributes) is sent to the renderer for processing. All other output commands (i.e. output primitives and structure output commands) are skipped. The traversal is flat meaning that the current pipeline attributes will not be saved when a structure in the path is executed. However, the current path offset is incremented for each output command that is encountered during the state accumulation.

DATA STRUCTURES
typedef XID PEXRenderer;
typedef XID PEXStructure;

typedef struct {
    PEXStructure structure;
    unsigned long offset;
} PEXElementRef;

ERRORS
BadPEXPath
The specified path is invalid.

BadPEXRenderer
The specified renderer resource identifier is invalid.

SEE ALSO PEXBeginRendering(3)
PEXCreateStructure(3)
PEXCreateRenderer(3)
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXAddToNameSet – Add Names to Name Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td>void PEXAddToNameSet(Display *display, XID resource_id, PEXOCRequestType req_type, unsigned long count, PEXName *names)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>display</td>
</tr>
<tr>
<td></td>
<td>resource_id</td>
</tr>
<tr>
<td></td>
<td>req_type</td>
</tr>
<tr>
<td></td>
<td>count</td>
</tr>
<tr>
<td></td>
<td>names</td>
</tr>
<tr>
<td>RETURNS</td>
<td>None</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>This function creates an output primitive attribute which adds the specified list of names to the current name set. If any name is outside the supported range, that name is ignored.</td>
</tr>
<tr>
<td>DATA STRUCTURES</td>
<td>typedef unsigned long PEXName;</td>
</tr>
<tr>
<td>ERRORS</td>
<td>BadPEXRenderer</td>
</tr>
<tr>
<td></td>
<td>BadPEXStructure</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>PEXCreateNameSet(3)</td>
</tr>
<tr>
<td></td>
<td>PEXRemoveFromNameSet(3)</td>
</tr>
<tr>
<td></td>
<td>PEXGetImpDepConstants(3)</td>
</tr>
</tbody>
</table>
### NAME
PEXAnnotationText – 3D Annotation Text Primitive

### SYNTAX
```c
void PEXAnnotationText(Display *display, XID resource_id, PEXOCRequestType req_type, PEXCoord *origin, PEXCoord *offset, int length, char *string)
```

### PARAMETERS
- **display**: A pointer to a display structure returned by a successful `XOpenDisplay` call.
- **resource_id**: The resource identifier of the renderer or structure.
- **req_type**: The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- **origin**: The origin of the text string.
- **offset**: The relative position of the text string from the origin.
- **length**: The number of bytes in the text string.
- **string**: A pointer to the text string.

### RETURNS
None

### DESCRIPTION
This function creates an annotation text output primitive. The first character set in the text font will be used.

The `origin` defines, in modeling coordinates, the position of the first character in the text string. The `offset` specifies an offset from the `origin` in normalized projection coordinates. The point where the text string is placed is called the annotation point. The annotation point is computed by adding the value of `offset` to the transformed `origin` point. The z-component of the annotation point specifies the x-y plane in normalized projection coordinate space on which the annotation text string is placed.

Depending on the text attribute source flag values, the text color, text precision, character expansion, character spacing, and text font index attributes are either obtained directly from the current text attribute values or from the entry in the text bundle specified by the current text bundle index attribute. The annotation text height, annotation text path, annotation text alignment, annotation text up vector, and annotation text style are also used to render the text string.

The annotation text string’s color is affected only by the depth-cueing computation (the reflectance stage of the rendering pipeline affects only surface primitives) and is mapped to a device color. The entire annotation text primitive is clipped if the `origin` is clipped. If the `origin` is not clipped, the annotation text is clipped according to text clipping rules. The connection line, if any, is clipped according to polyline clipping rules, except that modeling clipping is ignored. The current set of polyline attributes is used to render the connection line.

### DATA STRUCTURES
See PEXlib.h.

Modified November 1995
ERRORS

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO
PEXSetTextFontIndex(3)
PEXSetTextPrecision(3)
PEXSetCharExpansion(3)
PEXSetCharSpacing(3)
PEXSetTextColorIndex(3)
PEXSetTextColor(3)
PEXSetATextHeight(3)
PEXSetATextUpVector(3)
PEXSetATextPath(3)
PEXSetATextAlignment(3)
PEXSetATextStyle(3)
PEXSetTextBundleIndex(3)
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXAnnotationText2D – 2D Annotation Text Primitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td>void PEXAnnotationText2D(Display *display, XID resource_id, PEXOCRequestType req_type, PEXCoord2D *origin, PEXCoord2D *offset, int length, char *string)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>display A pointer to a display structure returned by a successful XOpenDisplay call.</td>
</tr>
<tr>
<td></td>
<td>resource_id The resource identifier of the renderer or structure.</td>
</tr>
<tr>
<td></td>
<td>req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).</td>
</tr>
<tr>
<td></td>
<td>origin The origin of the text string.</td>
</tr>
<tr>
<td></td>
<td>offset The relative position of the text string from the origin.</td>
</tr>
<tr>
<td></td>
<td>length The number of bytes in the text string.</td>
</tr>
<tr>
<td></td>
<td>string A pointer to the text string.</td>
</tr>
<tr>
<td>RETURNS</td>
<td>None</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>This function creates a 2D annotation text output primitive.</td>
</tr>
<tr>
<td></td>
<td>This primitive functions like PEXAnnotationText(3) except that the origin and offset positions are specified using only x- and y-components, and the z-component is always assumed to be zero. This primitive is two-dimensional only in that the z-component is implied, since geometry transformations are still carried out in three dimensions.</td>
</tr>
<tr>
<td>DATA STRUCTURES</td>
<td>See PEXlib.h.</td>
</tr>
<tr>
<td>ERRORS</td>
<td>BadPEXRenderer The specified renderer resource identifier is invalid.</td>
</tr>
<tr>
<td></td>
<td>BadPEXStructure The specified structure resource identifier is invalid.</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>PEXSetTextFontIndex(3)</td>
</tr>
<tr>
<td></td>
<td>PEXSetTextPrecision(3)</td>
</tr>
<tr>
<td></td>
<td>PEXSetCharExpansion(3)</td>
</tr>
<tr>
<td></td>
<td>PEXSetCharSpacing(3)</td>
</tr>
<tr>
<td></td>
<td>PEXSetTextColorIndex(3)</td>
</tr>
<tr>
<td></td>
<td>PEXSetTextColor(3)</td>
</tr>
<tr>
<td></td>
<td>PEXSetATextHeight(3)</td>
</tr>
<tr>
<td></td>
<td>PEXSetATextUpVector(3)</td>
</tr>
<tr>
<td></td>
<td>PEXSetATextPath(3)</td>
</tr>
<tr>
<td></td>
<td>PEXSetATextAlignment(3)</td>
</tr>
</tbody>
</table>

modified November 1995
PEXAnnotationText2D(3)  
PEXSetATextStyle(3)  
PEXSetTextBundleIndex(3)
NAME
PEXApplicationData – Structure Application Data

SYNTAX
void PEXApplicationData(Display *display, XID resource_id, PEXOCRequestType
req_type, int length, char *data)

PARAMETERS
display  A pointer to a display structure returned by a successful XOpenDisplay
call.
resource_id  The resource identifier of the renderer or structure.
req_type  The request type for the output command (PEXOCRender, PEXOC-
Store, PEXOCRenderSingle or PEXOCStoreSingle).
length  The length, in bytes, of the application data.
data  A pointer to the application data.

RETURNS
None

DESCRIPTION
This function creates an application data output command which has no visible effect
when rendered. It is typically used to store arbitrary application data in a structure.

ERRORS
BadPEXRenderer
The specified renderer resource identifier is invalid.
BadPEXStructure
The specified structure resource identifier is invalid.
NAME
PEXBeginPickAll – Begin Renderer Pick All

SYNTAX
void PEXBeginPickAll(Display *display, Drawable drawable, PEXRenderer renderer, long structure_id, int method, int send_event, int max_hits, int pick_device_type, PEXPickRecord *pick_record)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
drawable The resource identifier of a drawable.
renderer The resource identifier of the renderer.
structure_id A value to be used as the structure identifier for the root of the structure network.
method The pick all method (PEXPickAllAll or PEXPickAllVisible).
send_event TRUE or FALSE — specifying whether the server should send an event when the maximum number of hits is reached.
max_hits The maximum number of hits to be returned.
pick_device_type The pick device type (PEXPickDeviceDCHitBox or PEXPickDeviceNCHitVolume).
pick_record A pointer to the pick data record.

RETURNS
None

DESCRIPTION
This function starts an immediate-mode pick, setting the renderer’s renderer state to PEXPicking. When the renderer state is PEXPicking, primitives are hit tested instead of converted to pixels. All picked primitives are recorded until reaching the maximum hits specified is reached. Additional picked primitives will not be recorded. Once the the maximum number of hits is reached, subsequent primitives may be ignored.

The supported pick device types are inquirable via PEXGetEnumTypeInfo(3). The specified structure identifier will be inserted as the first structure component in the returned pick path(s).

If the send_event flag is True, and the pick method is PEXPickAllAll, then a PEXMaxHitsReached event is sent from the server to the client whenever the maximum number of hits is reached by the server, if the event is supported (see PEXGetImpDepConstants(3)). Upon receiving the event, the application should stop sending primitives and process the recorded hits. If the pick method is PEXPickAllVisible, a complete set of primitives must be sent to the server before determining which primitives are picked.

If the specified drawable does not have the same root and depth as the drawable used to create the renderer, or, if the specified drawable is not one of the supported drawables returned by PEXMatchRenderingTargets(3), a match error is generated. If the renderer state is set to PEXRendering or PEXPicking when this function is called, then the operation in progress is aborted, the PEXBeginPickAll function is completed, and a

modified November 1995
BadPEXRendererState error returned.

All functions which process output commands or manipulate attributes (i.e. all output command functions, PEXBeginStructure(3), PEXEndStructure(3), PEXRenderElements(3), and PEXAccumulateState(3)) can be called when the renderer state is PEXPicking. They will have the same semantics except that primitives are hit tested instead of converted to pixels.

data structures

typedef XID PEXRenderer;

typedef union {
  PEXPDNPCSubVolume volume;
  PEXPDDCHitBox box;
  PEXPickDataRecord data;
} PEXPickRecord;

typedef PEXNPCSubVolume PEXPDNPCHitVolume;

typedef struct {
  PEXCoord min;
  PEXCoord max;
} PEXNPCSubVolume;

typedef struct {
  float x;
  float y;
  float z;
} PEXCoord;

typedef struct {
  PEXDeviceCoord2D position;
  float distance;
} PEXPDDCHitBox;

typedef struct {
  short x;
  short y;
} PEXDeviceCoord2D;

typedef struct {
  unsigned short length; /* number of bytes in record */
  char *record;
} PEXPickDataRecord;

modified November 1995
typedef struct {
    int type;
    unsigned long serial; /* # of last request processed by server */
    Bool send_event; /* true if this came from a SendEvent request */
    Display *display; /* Display the event was read from */
    PEXRenderer renderer;
} PEXMaxHitsReachedEvent;

ERRORS
BadAlloc
The server failed to allocate resources necessary to complete request.

BadDrawable
The specified drawable resource identifier is invalid.

BadMatch
The specified drawable is unsupported, or the specified renderer resource was
not created with a compatible drawable.

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXRendererState
The specified renderer was in an invalid state.

BadValue
The pick record contains invalid data, or the pick device type is invalid.

SEE ALSO
PEXEndPickAll(3)
PEXPickAll(3)
PEXGetImpDepConstants(3)
### NAME
PEXBeginPickOne – Begin Renderer Pick One

### SYNTAX
```c
void PEXBeginPickOne(Display *display, Drawable drawable, PEXRenderer renderer,
long structure_id, int method, int pick_device_type, PEXPickRecord *pick_record)
```

### PARAMETERS
- **display**
  A pointer to a display structure returned by a successful `XOpenDisplay` call.
- **drawable**
  The resource identifier of a drawable.
- **renderer**
  The resource identifier of the renderer.
- **structure_id**
  A value to be used as the structure identifier for the root of the structure network.
- **method**
  The pick one method (PEXPickLast, PEXPickClosestZ, PEXPickVisibleAny, PEXPickVisibleClosest).
- **pick_device_type**
  The pick device type (PEXPickDeviceDCHitBox or PEXPickDeviceNPCHitVolume).
- **pick_record**
  A pointer to the pick data record.

### RETURNS
None

### DESCRIPTION
This function starts an immediate-mode pick, setting the renderer’s renderer state to PEXPicking. When the renderer state is PEXPicking, primitives are hit tested instead of converted to pixels. For pick one, a hierarchical path to the picked primitive will be maintained.

Standard pick one methods are PEXPickLast, PEXPickClosestZ, PEXPickVisibleAny and PEXPickVisibleClosest. The supported pick device types are inquirable via PEXGetEnumTypeInfo(3). The specified structure identifier will be inserted as the first structure component in the returned pick path.

If the specified drawable does not have the same root and depth as the drawable that was used to create the renderer, or if the specified drawable is not one of the supported drawables returned by PEXMatchRenderingTargets(3), a Match error will be generated. If the renderer state is set to PEXRendering or PEXPicking when this function is called, then the operation in progress is aborted, the PEXBeginPickOne function is completed, and a BadPEXRendererState error is sent.

All functions which process output commands or manipulate attributes (i.e. all output command functions, PEXBeginStructure(3), PEXEndStructure(3), PEXRenderElements(3), and PEXAccumulateState(3)) can be called when the renderer state is PEXPicking. They will have the same semantics except that primitives are hit tested instead of converted to pixels.
typedef XID PEXRenderer;

typedef union {
    PEXPDNPCHitVolume volume;
    PEXPDDCHitBox box;
    PEXPickDataRecord data;
} PEXPickRecord;

typedef PEXNPCSubVolume PEXPDNPCHitVolume;

typedef struct {
    PEXCoord min;
    PEXCoord max;
} PEXNPCSubVolume;

typedef struct {
    float x;
    float y;
    float z;
} PEXCoord;

typedef struct {
    PEXDeviceCoord2D position;
    float distance;
} PEXPDDCHitBox;

typedef struct {
    short x;
    short y;
} PEXDeviceCoord2D;

typedef struct {
    unsigned short length; /* number of bytes in record */
    char *record;
} PEXPickDataRecord;

ERRORS

BadAlloc
The server failed to allocate resources necessary to complete request.

BadDrawable
The specified drawable resource identifier is invalid.

BadMatch
The specified drawable is unsupported, or the specified renderer resource was not created with a compatible drawable.
BadPEXRenderer
   The specified renderer resource identifier is invalid.

BadPEXRendererState
   The specified renderer was in an invalid state.

BadValue
   The pick record contains invalid data, or the pick device type is invalid.

SEE ALSO
   PEXEndPickOne(3)
   PEXPickOne(3)
NAME  PEXBeginRendering – Begin Rendering

SYNTAX  void PEXBeginRendering(Display *display, Drawable drawable, PEXRenderer renderer)

PARAMETERS  
- display  A pointer to a display structure returned by a successful XOpenDisplay call.
- drawable  The resource identifier of a drawable.
- renderer  The resource identifier of a renderer.

RETURNS  None

DESCRIPTION  This function initializes the specified renderer and binds the specified drawable to it. Subsequent output primitive commands sent to renderer produce output to the specified drawable. The renderer’s pipeline state is initialized to the values in its pipeline context, or to default values if there is no pipeline context. This function causes the renderer’s renderer state to be set to PEXRendering and its current path to be set to <0,0>. The renderer’s HLHSR mode is used to initialize hidden surface computations. (For example, the z-buffer is initialized if the HLHSR mode is set to PEXHLHSRZBuffer or PEXHLHSRZBufferID.) If the renderer’s state is PEXRendering or PEXPicking when this request is received, an implicit end rendering or picking request is performed before the begin rendering request is executed. Output commands received by a renderer are ignored if the state is not PEXRendering or PEXPicking.

DATA STRUCTURES  
- typedef XID PEXRenderer;

ERRORS  
- BadAlloc
  The server failed to allocate the resource.
- BadDrawable
  The specified drawable resource identifier is invalid.
- BadMatch
  The specified drawable is unsupported, or the specified renderer resource was not created with a compatible drawable.
- BadPEXRenderer
  The specified renderer resource identifier is invalid.
- BadPEXRendererState
  The specified renderer was in an invalid state.

SEE ALSO  PEXCreateRenderer(3)  PEXEndRendering(3)

modified November 1995
NAME
PEXBeginStructure – Save Rendering Pipeline State

SYNTAX
void PEXBeginStructure(Display *display, PEXRenderer renderer, long structure_id)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
renderer The resource identifier of a renderer.
structure_id A value to be used as an application-specified structure identifier.

RETURNS
None

DESCRIPTION
Application programs can use this function to simulate the effects of the execute structure output command. This request saves the rendering pipeline attributes in the specified renderer. This request also increments the element offset of the last entry in the renderer’s current path. The function then adds the structure id and an element offset of zero to the renderer’s current path.

The rendering pipeline’s global transform attribute is set to the matrix computed by concatenating the current local transform and current global transform matrices. The local transform matrix is then set to the identity matrix.

DATA STRUCTURES
typedef XID PEXRenderer;

ERRORS
BadPEXRenderer
The specified renderer resource identifier is invalid.

SEE ALSO
PEXEndStructure(3)
PEXExecuteStructure(3)
PEXCreateRenderer(3)

modified November 1995
NAME  PEXBuildTransform – utility function

SYNTAX  void PEXBuildTransform(PEXCoord *fixed_point, PEXVector *trans_vector, double angle_x, double angle_y, double angle_z, PEXVector *scale_vector, PEXMatrix matrix_return)

PARAMETERS
fixed_point  Origin for scaling and rotation.
trans_vector  Translation vector.
angle_x  Angle of rotation about X axis, in radians.
angle_y  Angle of rotation about Y axis, in radians.
angle_z  Angle of rotation about Z axis, in radians.
scale_vector  Vector of scale factors for X, Y and Z.
matrix_return  Matrix into which result is stored.

RETURNS  None

DESCRIPTION  This function builds a transformation matrix that scales by the values in the scale vector about the fixed point, rotates about the X, Y and Z axes using the fixed point as the center of rotation and then translates according to translation vector, in that order.

ERRORS  None

SEE ALSO  PEXBuildTransform2D(3)
NAME PEXBuildTransform2D – utility function

SYNTAX

void PEXBuildTransform2D(PEXCoord2D *fixed_point, PEXVector2D *trans_vector, double angle_z, PEXVector2D *scale_vector, PEXMatrix3x3 *matrix_return)

PARAMETERS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fixed_point</td>
<td>Origin for scaling and rotation.</td>
</tr>
<tr>
<td>trans_vector</td>
<td>Translation vector.</td>
</tr>
<tr>
<td>angle_z</td>
<td>Angle of rotation about Z axis, in radians.</td>
</tr>
<tr>
<td>scale_vector</td>
<td>Vector of scale factors for X and Y axes.</td>
</tr>
<tr>
<td>matrix_return</td>
<td>Matrix into which result is stored.</td>
</tr>
</tbody>
</table>

RETURNS None

DESCRIPTION This function builds a 3X3 transformation matrix that scales by the values in the scale vector about the fixed point, rotates about Z axis using the fixed point as the center of rotation and then translates according to translation vector, in that order.

ERRORS None

SEE ALSO PEXBuildTransform(3)
NAME
PEXCellArray – 3D Cell Array Primitive

SYNTAX
void PEXCellArray(Display ∗display, XID resource_id, PEXOCRequestType req_type,
PEXCoord ∗point1, PEXCoord ∗point2, PEXCoord ∗point3, unsigned int col_count,
unsigned int row_count, PEXTableIndex ∗color_indices)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay
call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOC-
Store, PEXOCRenderSingle or PEXOCStoreSingle).
point1 The first cell array definition point.
point2 The second cell array definition point.
point3 The third cell array definition point.
col_count The number of cell columns (number of cells in the X direction).
row_count The number of cell rows (number of cells in the Y direction).
color_indices An array of color table index values which specify the color of each cell.

RETURNS
None

DESCRIPTION
This function creates a 3D cell-array output primitive.
A 3D cell-array primitive is a parallelogram of equally-sized cells, each of which is a
parallelogram with a single color. Each cell has a width defined by:

\[
width = \sqrt{\left(\text{point}_1 \to x - \text{point}_2 \to x\right)^2 + \left(\text{point}_1 \to y - \text{point}_2 \to y\right)^2 + \left(\text{point}_1 \to z - \text{point}_2 \to z\right)^2} / \text{col_count}
\]

and a height defined by:

\[
height = \sqrt{\left(\text{point}_1 \to x - \text{point}_3 \to x\right)^2 + \left(\text{point}_1 \to y - \text{point}_3 \to y\right)^2 + \left(\text{point}_1 \to z - \text{point}_3 \to z\right)^2} / \text{row_count}
\]

Cell colors are specified in a one-dimensional array where the colors are stored in row-
major order. The first color in the array is the color at the cell at the corner of point1, and
subsequent colors represent the colors of cells proceeding to point2.
If any color index is not defined, color index one is used. If color index one is not
defined, the resulting color is implementation-dependent. The column count and row
count can not be zero.

DATA STRUCTURES
typedef unsigned short PEXTableIndex;
See also PEXlib.h.

20 modified November 1995
<table>
<thead>
<tr>
<th>ERRORS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BadPEXOutputCommand</td>
<td>The output command contains an invalid value.</td>
</tr>
<tr>
<td>BadPEXRenderer</td>
<td>The specified renderer resource identifier is invalid.</td>
</tr>
<tr>
<td>BadPEXStructure</td>
<td>The specified structure resource identifier is invalid.</td>
</tr>
</tbody>
</table>
PEXCellArray2D – 2D Cell Array Primitive

void PEXCellArray2D(Display *display, XID resource_id, PEXOCRequestType req_type, PEXCoord2D *point1, PEXCoord2D *point2, unsigned int col_count, unsigned int row_count, PEXTableIndex *color_indices)

A pointer to a display structure returned by a successful XOpenDisplay call.

The resource identifier of the renderer or structure.

The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).

The first cell array definition point.

The second cell array definition point.

The number of cell columns (number of cells in the X direction).

The number of cell rows (number of cells in the Y direction).

An array of color table index values which specify the color of each cell.

None

This function creates a 2D cell-array output primitive.

A 2D cell-array is a rectangle of equally-sized cells, each of which is a rectangle which has a single color. The primitive is specified by two points which define a rectangle aligned with the modeling coordinate axes. The width and height of each cell is defined as in PEXCellArray(3) where the z component is zero.

All other aspects of this primitive are the same as PEXCellArray(3).

typedef unsigned short PEXTableIndex;

See also PEXlib.h.

BadPEXOutputCommand

The output command contains an invalid value.

BadPEXRenderer

The specified renderer resource identifier is invalid.

BadPEXStructure

The specified structure resource identifier is invalid.

modified November 1995
NAME
PEXChangeNameSet – Change Name Set

SYNTAX
void PEXChangeNameSet(Display *display, PEXNameSet nameset, int action, unsigned long count, PEXName *names)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
nameset The resource identifier of the name set.
action Type of change to be made (PEXNSAdd, PEXNSRemove, PEXNSReplace).
count The number of names.
names An array of names.

RETURNS
None

DESCRIPTION
This function modifies the specified name set resource. The list of names is added to the name set if the specified action is PEXNSAdd, removed from the name set if the specified action is PEXNSRemove, or used to replace the names currently in the name set if the specified action is PEXNSReplace. If requested to remove a name that does not exist, it is silently ignored.

DATA STRUCTURES
typedef XID PEXNameSet;
typedef unsigned long PEXName;

ERRORS
BadPEXNameSet
The specified name set resource identifier is invalid.
BadValue
The specified value for action parameter is invalid.

SEE ALSO
PEXCreateNameSet(3)
PEXGetNameSet(3)
NAME PEXChangePickDevice – Change Pick Device Attributes

SYNTAX

void PEXChangePickDevice(Display ∗display, PEXWorkstation workstation, int
pick_device_type, unsigned long value_mask, PEXPDAtributes ∗values)

PARAMETERS

display A pointer to a display structure returned by a successful XOpenDisplay
call.
workstation The resource identifier of the workstation.

pick_device_type The pick device type (PEXPickDeviceDCHitBox or PEXPickDeviceNPC-
CHitVolume).

value_mask A mask indicating which attributes to return.
values A pointer to the pick device attribute values.

RETURNS None

DESCRIPTION

This function will modify the attributes of a pick descriptor for the PHIGS workstation
resource specified. The descriptor to be modified will be the currently-defined descriptor
for the pick device of the type specified. Supported pick device types are inquirable via
PEXGetEnumTypeInfo(3). The value mask indicates which attributes are to be changed.
The value mask is constructed by OR’ing together the following constants:

PEXPDPickStatus
PEXPDPickPath
PEXPDPickPathOrder
PEXPDPickIncl
PEXPDPickExcl
PEXPDPickDataRec
PEXPDPromptEchoType
PEXPDEchoVolume
PEXPDEchoSwitch

DATA

STRUCTURES

typedef XID PEXWorkstation;

typedef struct {
    unsigned short status;
    PEXPickPath path;
    int path_order;
    PEXNameSet inclusion;
    PEXNameSet exclusion;
    PEXPickRecord pick_record;
    PEXEnumTypeIndex prompt_echo_type;
    PEXViewport echo_volume;
    int echo_switch;
} PEXPDAttributes;

modified November 1995
typedef struct {
    unsigned long count;       /* number of elements */
    PEXPickElementRef *elements;
} PEXPickPath;

typedef struct {
    PEXStructure sid;
    unsigned long offset;
    unsigned long pick_id;
} PEXPickElementRef;

typedef XID PEXStructure;

typedef XID PEXNameSet;

typedef union {
    PEXPDNPCSubVolume volume;
    PEXPDDCHitBox box;
    PEXPickDataRecord data;
} PEXPickRecord;

typedef PEXNPCSubVolume PEXPDNPCSubVolume;

typedef struct {
    PEXCoord min;
    PEXCoord max;
} PEXNPCSubVolume;

typedef struct {
    float x;
    float y;
    float z;
} PEXCoord;

typedef struct {
    PEXDeviceCoord2D position;
    float distance;
} PEXPDDCHitBox;

typedef struct {
    short x;
    short y;
} PEXDeviceCoord2D;
typedef struct {
    unsigned short length; /* number of bytes in record */
    char *record;
} PEXPickDataRecord;

typedef short PEXEnumTypeIndex;

typedef struct {
    PEXDeviceCoord min;
    PEXDeviceCoord max;
    PEXSwitch use_drawable;
    unsigned char reserved[3];
} PEXViewport;

typedef struct {
    short x;
    short y;
    float z;
} PEXDeviceCoord;

typedef unsigned char PEXSwitch;

**ERRORS**

**BadPEXNameSet**
The specified name set resource identifier is invalid.

**BadPEXPath**
The specified path is invalid.

**BadPEXWorkstation**
The specified *workstation* resource identifier is invalid.

**BadValue**
The specified pick device type is invalid, a specified value is invalid, or an invalid bit set in the value mask.

**SEE ALSO**
PEXGetPickDevice(3)
PEXGetEnumTypeInfo(3)
NAME
PEXChangePipelineContext – Change Pipeline Context

SYNTAX
void PEXChangePipelineContext(Display *display, PEXPipelineContext context, unsigned long *value_mask, PEXPCAttributes *values)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
context The resource identifier of the pipeline context.
value_mask A pointer to an array of three unsigned long.
values A pointer to new values for attributes in the pipeline context.

RETURNS
None

DESCRIPTION
This function changes components of the specified pipeline context to the values specified. The value mask indicates which attribute values are specified. PEXSetPCAttributeMask(3) and PEXSetPCAttributeMaskAll(3) can be called to setup the value mask.

DATA STRUCTURES
typedef XID PEXPipelineContext;
See also PEXlib.h.

ERRORS
BadPEXColorType
The specified color type is invalid or unsupported.

BadPEXNameSet
The specified name set resource identifier is invalid.

BadPEXPipelineContext
The specified pipeline context resource identifier is invalid.

BadValue
A specified value is out of range, or an invalid bit is set in the value mask.

SEE ALSO
PEXCreatePipelineContext(3)
PEXGetPipelineContext(3)
PEXSetPCAttributeMask(3)
NAME
PEXChangeRenderer – Change Renderer Attributes

SYNTAX
void PEXChangeRenderer(Display *display, PEXRenderer renderer, unsigned long value_mask, PEXRendererAttributes *values)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
renderer The resource identifier of the renderer.
value_mask A mask indicating the renderer attributes to be changed.
values A pointer to new values for the renderer attributes.

RETURNS
None

DESCRIPTION
This function modifies attributes of the specified renderer resource. The value mask indicates the attribute values to be modified. The value mask is constructed by OR’ing together the following constants:

- PEXRABackgroundColor
- PEXRAInvisibilityIncl
- PEXRAClearImage
- PEXRALightTable
- PEXRAClearZ
- PEXRALineBundle
- PEXRAClipList
- PEXRAMarkerBundle
- PEXRAColorApproxTable
- PEXRANPCSubVolume
- PEXRAColorTable
- PEXRAPatternTable
- PEXRADepthCueTable
- PEXRAPickExcl
- PEXRAEchoMode
- PEXRAPickIncl
- PEXRAEdgeBundle
- PEXRAPickStartPath
- PEXRAHLHSRMode
- PEXRAPipelineContext
- PEXRAHighlightExcl
- PEXRATextBundle
- PEXRAHighlightIncl
- PEXRATextFontTable
- PEXRAInteriorBundle
- PEXRAViewTable
- PEXRAInvisibilityExcl
- PEXRAViewport

Note that the renderer’s current path and renderer state attributes cannot be set explicitly using this function.

DATA STRUCTURES
typedef XID PEXRenderer;

See also PEXlib.h.

ERRORS
BadMatch
The specified lookup table resource was not created with a drawable compatible with the drawable used to create the renderer resource.

BadPEXLookupTable
The specified lookup table resource identifier is invalid.
The specified name set resource identifier is invalid.

BadPEXPipelineContext
The specified pipeline context resource identifier is invalid.

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadValue
A specified value is out of range, or an invalid bit is set in the value mask.

SEE ALSO
PEXCreateRenderer(3)
PEXGetRendererAttributes(3)
PEXGetRendererDynamics(3)
NAME   PEXChangeSearchContext – Change Search Context

SYNTAX  void PEXChangeSearchContext(Display *display, PEXSearchContext context, unsigned long value_mask, PEXSCAttributes *values)

PARAMETERS  display A pointer to a display structure returned by a successful XOpenDisplay call.
context The resource identifier of the search context.
value_mask A mask indicating the search context attributes to be changed.
values A pointer to new values for the specified search context attributes.

RETURNS None

DESCRIPTION This function modifies the attributes of the specified search context resource. The value mask indicates which values are specified. The value mask is constructed by OR’ing together the following constants:

PEXSCPosition
PEXSCDistance
PEXSCCeiling
PEXSCModelClipFlag
PEXSCStartPath
PEXSCNormalList
PEXSCInvertedList

DATA STRUCTURES typedef XID PEXSearchContext;

typedef struct {
    PEXCoord position;
    float distance;
    unsigned short ceiling;
    Bool model_clip_flag;
    PEXStructurePath start_path;
    PEXListOfNameSetPair normal;
    PEXListOfNameSetPair inverted;
} PEXSCAttributes;

typedef struct {
    float x;
    float y;
    float z;
} PEXCoord;

modified November 1995
typedef struct {
    unsigned long count; /* number of elements */
    PEXElementRef *elements;
} PEXStructurePath;

typedef struct {
    PEXStructure structure;
    unsigned long offset;
} PEXElementRef;

typedef XID PEXStructure;

typedef struct {
    unsigned short count; /* number of pairs */
    PEXNameSetPair *pairs;
} PEXListOfNameSetPair;

typedef struct {
    PEXNameSet inclusion;
    PEXNameSet exclusion;
} PEXNameSetPair;

typedef XID PEXNameSet;

ERRORS

BadPEXNameSet
The specified name set resource identifier is invalid.

BadPEXPath
The specified path is invalid.

BadPEXSearchContext
The specified search context resource identifier is invalid.

BadValue
A specified value is out of range, or an invalid bit is set in the value mask.

SEE ALSO
PEXCreateSearchContext(3)
PEXGetSearchContext(3)
NAME
PEXChangeStructureRefs – Change Structure References

SYNTAX
void PEXChangeStructureRefs(Display *display, PEXStructure old_structure, PEXStructure new_structure)

PARAMETERS
display  A pointer to a display structure returned by a successful XOpenDisplay call.
old_structure  The resource identifier of the structure no longer to be referenced.
new_structure  The resource identifier of the structure now referenced.

RETURNS
None

DESCRIPTION
This function changes execute structure elements in the server that reference the specified old structure into execute structure elements which reference the specified new structure. Both structures must already exist as valid structure resources.
Any references to the new structure that existed before this request are not affected. If there were references to the old structure and the new structure does not exist, an error is returned and no action is taken.
On all PHIGS workstation resources where the new structure is not already posted and the old structure is posted, the new structure is posted with the same priority as the old structure and the old structure is unposted.

DATA STRUCTURES
typedef XID PEXStructure;

ERRORS
BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO
PEXCreateStructure(3)
PEXGetStructureInfo(3)
PEXGetStructuresInNetwork(3)
PEXGetAncestors(3)
PEXGetDescendants(3)
NAME

PEXObjectBytesToOC – Copy Encoded Output Commands

SYNTAX

void PEXCopyBytesToOC(Display *display, int length, char *data)

PARAMETERS

display A pointer to a display structure returned by a successful XOpenDisplay call.
length The number of bytes to copy.
data A pointer to the output command data.

RETURNS

None

DESCRIPTION

This function copies the specified number of bytes of data to the transport buffer. It is recommended that the number of bytes be a multiple of four as the protocol format requires output commands to be aligned on four-byte boundaries. It is the application’s responsibility to ensure that alignment restrictions are met. PEXStartOCs(3) must be called prior to this.

ERRORS

None

SEE ALSO

PEXObjectBytesToOC(3)  
PEXObjectStartOCs(3)  
PEXObjectFinishOCs(3)  
PEXObjectGetOCAddr(3)  

modified November 1995
NAME  PEXCopyElements – Copy Elements

SYNTAX  void PEXCopyElements(Display *display, PEXStructure src_structure, int src_whence1, long src_offset1, int src_whence2, long src_offset2, PEXStructure dst_structure, int dst_whence, long dst_offset)

PARAMETERS  

  display  A pointer to a display structure returned by a successful XOpenDisplay call.
  src_structure  The resource identifier of the source structure.
  src_whence1  A value specifying, with src_offset1, the first limit of the range of elements to be copied (PEXBeginning, PEXCurrent, PEXEnd).
  src_offset1  The offset from src_whence1 denoting the first limit of the range of elements to be copied.
  src_whence2  A value specifying, with src_offset2, the second limit of the range of elements to be copied (PEXBeginning, PEXCurrent, PEXEnd).
  src_offset2  The offset from src_whence2 denoting the second limit of the range of elements to be copied.
  dst_structure  The resource identifier of the destination structure.
  dst_whence  A value specifying, with dst_offset, the position at which the elements are inserted into the destination structure (PEXBeginning, PEXCurrent, PEXEnd).
  dst_offset  The offset from dst_whence denoting the position at which the elements are inserted into the destination structure.

RETURNS  None

DESCRIPTION  This function copies a range of elements from the specified source structure to the specified destination structure.

If a computed offset is less than zero, it is set to zero before obtaining the element information. If a computed offset is greater than the number of elements in the structure, the offset is set to the offset of the last element in the structure.

The source structure and destination structure can be the same. In this case, the copy operation proceeds as though the indicated range were copied to a temporary location and then inserted relative to the destination position.

After the copy operation, the element pointer of the destination structure is updated to point at the last element copied into the destination structure. The editing mode attribute of the destination structure is ignored during this request. The copied elements are always inserted into the destination structure and are never used to replace existing structure elements.
### DATA STRUCTURES

typedef XID PEXStructure;

### ERRORS

- **BadPEXStructure**
  The specified structure resource identifier is invalid.
- **BadValue**
  The specified value for whence parameter is invalid.

### SEE ALSO

- PEXCreateStructure(3)
NAME
PEXCcopyLookupTable – Copy Lookup Table

SYNTAX
void PEXCopyLookupTable(Display *display, PEXLookupTable src_table, PEXLookupTable dst_table)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
src_table The resource identifier of the source lookup table.
dst_table The resource identifier of the destination lookup table.

RETURNS None

DESCRIPTION
This function copies entries from the source lookup table to destination lookup table, after first deleting all entries in the destination lookup table. Both tables must already exist as valid lookup table resources. Both must have been created for the same class of drawables, and both must be the same table type.

DATA STRUCTURES
typedef XID PEXLookupTable;

ERRORS
BadMatch The specified lookup tables were not created with compatible drawables.
BadPEXLookupTable A specified lookup table resource identifier is invalid, or the table type is unsupported.

SEE ALSO PEXCreateLookupTable(3)
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXCopyNameSet – Copy Name Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td>void PEXCopyNameSet(Display *display, PEXNameSet src_nameset, PEXNameSet dst_nameset)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>display</td>
</tr>
<tr>
<td></td>
<td>src_nameset</td>
</tr>
<tr>
<td></td>
<td>dst_nameset</td>
</tr>
<tr>
<td>RETURNS</td>
<td>None</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>This function copies the contents of the source name set to the destination name set. Both must already exist as valid name set resources. All values already in the destination name set are overwritten.</td>
</tr>
<tr>
<td>DATA STRUCTURES</td>
<td>typedef XID PEXNameSet;</td>
</tr>
<tr>
<td>ERRORS</td>
<td>BadPEXNameSet</td>
</tr>
<tr>
<td></td>
<td>The specified name set resource identifier is invalid.</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>PEXCreateNameSet(3)</td>
</tr>
</tbody>
</table>

modified November 1995
NAME  PEXCopyPipelineContext – Copy Pipeline Context

SYNTAX  void PEXCopyPipelineContext(Display *display, unsigned long *value_mask, PEXPipelineContext src_context, PEXPipelineContext dst_context)

PARAMETERS  display  A pointer to a display structure returned by a successful XOpenDisplay call.
  value_mask  A pointer to an array of three unsigned long.
  src_context  The resource identifier of the source pipeline.
  dst_context  The resource identifier of the destination pipeline context.

RETURNS  None

DESCRIPTION  This function copies attributes from the source pipeline context to the destination pipeline context. Both must already exist as valid pipeline context resources. Only the attributes indicated by the value mask are copied and the remainder of the attributes are left as they were. PEXSetPCAttributeMask(3) and PEXSetPCAttributeMaskAll(3) can be called to setup the value mask.

DATA STRUCTURES  typedef XID PEXPipelineContext;

ERRORS  BadPEXPipelineContext  A specified pipeline context resource identifier is invalid.
  BadValue  An invalid bit is set in the value mask.

SEE ALSO  PEXCreatePipelineContext(3)
           PEXSetPCAttributeMask(3)

modified November 1995
**NAME**  
PEXCopySearchContext – Copy Search Context

**SYNTAX**  
```c  
void PEXCopySearchContext(Display *display, unsigned long value_mask, PEXSearchContext src_context, PEXSearchContext dst_context)  
```

**PARAMETERS**  
- `display` A pointer to a display structure returned by a successful `XOpenDisplay` call.
- `value_mask` A mask specifying which attributes are to be copied.
- `src_context` The resource identifier of the source search context.
- `dst_context` The resource identifier of the destination search context.

**RETURNS**  
None

**DESCRIPTION**  
This function copies attributes from the source search context to the destination search context resource. Both must already exist as valid search context resources. Attributes indicated by the value mask are copied and the remainder of the attributes are left unchanged. The value mask is constructed by OR'ing together the following constants:

- `PEXSCPosition`
- `PEXSCDistance`
- `PEXSCCeiling`
- `PEXSCModelClipFlag`
- `PEXSCStartPath`
- `PEXSCNormalList`
- `PEXSCInvertedList`

**DATA STRUCTURES**

```c  
typedef XID PEXSearchContext;  
```

**ERRORS**
- `BadPEXSearchContext`  
  A specified search context resource identifier is invalid.
- `BadValue`  
  An invalid bit is set in the value mask.

**SEE ALSO**  
PEXCreateSearchContext(3)

modified November 1995
### NAME
PEXCopystucture – Copy Structure

### SYNTAX
```c
void PEXCopyStructure(Display *display, PEXStructure src_structure, PEXStructure dst_structure)
```

### PARAMETERS
- **display**: A pointer to a display structure returned by a successful `XOpenDisplay` call.
- **src_structure**: The resource identifier of the source structure.
- **dst_structure**: The resource identifier of the destination structure.

### RETURNS
None

### DESCRIPTION
This function copies elements in the source structure to the destination structure. Both structures must already exist as valid structure resources. Elements already in the destination structure are overwritten. The element pointer and editing mode attributes of the source structure are copied to the destination structure as well.

### DATA STRUCTURES
```c
typedef XID PEXStructure;
```

### ERRORS
- **BadPEXStructure**: The specified structure resource identifier is invalid.

### SEE ALSO
PEXCreateStructure(3)
NAME
PEXCountOCs – Count Output Commands in an Encoded List

SYNTAX
unsigned long PEXCountOCs(int float_format, unsigned long length, char *encoded_ocs)

PARAMETERS
float_format The floating point format of the encoded output commands (PEX-IEEE_754_32, PEXDEC_F_Floating, PEXIEEE_754_64, PEXDEC_D_Floating).
length The length, in bytes, of the encoded output commands.
encoded_ocs A pointer to the encoded output commands.

RETURNS
The number of output commands represented in the encoded output commands.

DESCRIPTION
This function has no visible effect. This function returns the number of output commands in the encoded list of output commands.
A count of zero will be returned if the specified floating point format is not supported.

ERRORS
None

SEE ALSO
PEXDecodeOCs(3)
NAME
PEXCreateLookupTable – Create Lookup Table

SYNTAX
PEXLookupTable PEXCreateLookupTable(Display *display, Drawable drawable, int table_type)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
drawable The resource identifier of a drawable.
table_type The type of lookup table to be created (see the Description section).

RETURNS
The resource identifier of the newly-created lookup table.

DESCRIPTION
This function creates a lookup table resource of the specified type and returns the resource identifier of the new lookup table. The returned identifier is used to refer to the created lookup table. The type of lookup table to be created must be one of the following:

PEXLTColorApprox
PEXLTColor
PEXLTDepthCue
PEXLETedgeBundle
PEXLETinteriorBundle
PEXLETlight
PEXLETlineBundle
PEXLETMarkerBundle
PEXLETPattern
PEXLETTextBundle
PEXLETTextFont
PEXLETView

The newly-created lookup table can only be used with drawables having the same depth and root as the specified drawable.

DATA STRUCTURES
typedef XID PEXLookupTable;

ERRORS
BadAlloc
The server failed to allocate the resource.
BadDrawable
The specified drawable resource identifier is invalid.
BadMatch
The specified drawable is unsupported.
BadPEXLookupTable
The specified table type is unsupported.
BadValue

modified November 1995
The specified table type is invalid.

SEE ALSO

PEXFreeLookupTable(3)
### NAME
PEXCreateNameSet – Create Name Set

### SYNTAX
PEXNameSet PEXCreateNameSet(Display *display)

### PARAMETERS
- **display**: A pointer to a display structure returned by a successful XOpenDisplay call.

### RETURNS
The resource identifier of the newly-created name set.

### DESCRIPTION
This function creates a name set resource and returns the resource identifier of the created name set. The returned identifier is used to refer to the created name set.

### DATA STRUCTURES
```c
typedef XID PEXNameSet;
```

### ERRORS
- **BadAlloc**: The server failed to allocate the resource.

### SEE ALSO
PEXFreeNameSet(3)
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXCreatePickMeasure – Create Pick Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td>PEXPickMeasure PEXCreatePickMeasure(Display *display, PEXWorkstation workstation, int pick_device_type)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>A pointer to a display structure returned by a successful XOpenDisplay call.</td>
</tr>
<tr>
<td></td>
<td>The resource identifier of the workstation.</td>
</tr>
<tr>
<td></td>
<td>The pick device type (PEXPickDeviceDCHitBox or PEXPickDeviceNPCHitVolume).</td>
</tr>
<tr>
<td>RETURNS</td>
<td>The resource identifier of the newly-created pick measure.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>This function creates a pick measure resource of the type specified. The pick measure is initialized with the values contained in the appropriate pick device descriptor stored in the specified workstation. The supported pick device types are inquirable via PEXGetEnumTypeInfo(3).</td>
</tr>
<tr>
<td>DATA STRUCTURES</td>
<td>typedef XID PEXWorkstation;</td>
</tr>
<tr>
<td></td>
<td>typedef XID PEXPickMeasure;</td>
</tr>
<tr>
<td>ERRORS</td>
<td>BadAlloc</td>
</tr>
<tr>
<td></td>
<td>The server failed to allocate the resource.</td>
</tr>
<tr>
<td></td>
<td>BadPEXWorkstation</td>
</tr>
<tr>
<td></td>
<td>The specified workstation resource identifier is invalid.</td>
</tr>
<tr>
<td></td>
<td>BadValue</td>
</tr>
<tr>
<td></td>
<td>The specified pick device type is invalid.</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>PEXFreePickMeasure(3)</td>
</tr>
<tr>
<td></td>
<td>PEXGetEnumTypeInfo(3)</td>
</tr>
</tbody>
</table>
NAME
PEXCreatePipelineContext – Create Pipeline Context

SYNTAX
PEXPipelineContext PEXCreatePipelineContext(Display *display, unsigned long *value_mask, PEXPCAttributes *values)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
value_mask A pointer to an array of three unsigned long.
values A pointer to values to override default attribute values in the new pipeline context.

RETURNS
The resource identifier of the newly-created pipeline context.

DESCRIPTION
This function creates a pipeline context and returns its resource identifier. The value mask indicates the values specified. Attribute values not specified will be initialized to the default values shown in the table below. PEXSetPCAttributeMask(3) and PEXSetPCAttributeMaskAll(3) can be called to setup the value mask.

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEXPCMarkerType</td>
<td>PEXMarkerAsterisk</td>
</tr>
<tr>
<td>PEXPCMarkerScale</td>
<td>1.0</td>
</tr>
<tr>
<td>PEXPCMarkerColor</td>
<td>{PEXColorTypeIndexed, 1}</td>
</tr>
<tr>
<td>PEXPCMarkerBundleIndex</td>
<td>1</td>
</tr>
<tr>
<td>PEXPCTextFont</td>
<td>1</td>
</tr>
<tr>
<td>PEXPCTextPrecision</td>
<td>PEXStringPrecision</td>
</tr>
<tr>
<td>PEXPCCharExpansion</td>
<td>1.0</td>
</tr>
<tr>
<td>PEXPCCharSpacing</td>
<td>0.0</td>
</tr>
<tr>
<td>PEXPCTextColor</td>
<td>{PEXColorTypeIndexed, 1}</td>
</tr>
<tr>
<td>PEXPCCharHeight</td>
<td>0.01</td>
</tr>
<tr>
<td>PEXPCCharUpVector</td>
<td>&lt;0.0, 1.0&gt;</td>
</tr>
<tr>
<td>PEXPCPath</td>
<td>PEXPathRight</td>
</tr>
<tr>
<td>PEXPCAlignment</td>
<td>[PEXHAlignNormal, PEXVAlignNormal]</td>
</tr>
<tr>
<td>PEXPCTextHeight</td>
<td>0.01</td>
</tr>
<tr>
<td>PEXPCTextUpVector</td>
<td>&lt;0.0, 1.0&gt;</td>
</tr>
<tr>
<td>PEXPCPath</td>
<td>PEXPathRight</td>
</tr>
<tr>
<td>PEXPCAlignment</td>
<td>[PEXHAlignNormal, PEXVAlignNormal]</td>
</tr>
<tr>
<td>PEXPCTextStyle</td>
<td>PEXATextNotConnected</td>
</tr>
<tr>
<td>PEXPCBundleIndex</td>
<td>1</td>
</tr>
<tr>
<td>PEXPCLineType</td>
<td>PEXLineTypeSolid</td>
</tr>
<tr>
<td>PEXPCLineWidth</td>
<td>1.0</td>
</tr>
<tr>
<td>PEXPCColor</td>
<td>{PEXColorTypeIndexed, 1}</td>
</tr>
<tr>
<td>PEXPCCurveApprox</td>
<td>{1, 1.0}</td>
</tr>
<tr>
<td>PEXPCPolylineInterp</td>
<td>PEXPolylineInterpNone</td>
</tr>
<tr>
<td>PEXPCBundleIndex</td>
<td>1</td>
</tr>
<tr>
<td>PEXPCInteriorStyle</td>
<td>PEXInteriorStyleHollow</td>
</tr>
</tbody>
</table>

modified November 1995
<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEXPCInteriorStyleIndex</td>
<td>1</td>
</tr>
<tr>
<td>PEXPCSurfaceColor</td>
<td>{PEXColorTypeIndexed, 1}</td>
</tr>
<tr>
<td>PEXPCReflectionAttr</td>
<td>{1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, {PEXColorTypeIndexed, 1}}</td>
</tr>
<tr>
<td>PEXPCReflectionModel</td>
<td>PEXReflectionNone</td>
</tr>
<tr>
<td>PEXPCSurfaceInterp</td>
<td>PEXSurfaceInterpNone</td>
</tr>
<tr>
<td>PEXPCBFIinteriorStyle</td>
<td>PEXInteriorStyleHollow</td>
</tr>
<tr>
<td>PEXPCBFIinteriorStyleIndex</td>
<td>1</td>
</tr>
<tr>
<td>PEXPCBFSurfaceColor</td>
<td>{PEXColorTypeIndexed, 1}</td>
</tr>
<tr>
<td>PEXPCBFFreflectionAttr</td>
<td>{1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, {PEXColorTypeIndexed, 1}}</td>
</tr>
<tr>
<td>PEXPCBFFreflectionModel</td>
<td>PEXReflectionNone</td>
</tr>
<tr>
<td>PEXPCBFSurfaceInterp</td>
<td>PEXSurfaceInterpNone</td>
</tr>
<tr>
<td>PEXPCSurfaceApprox</td>
<td>{1, 1.0, 1.0, 0.0}</td>
</tr>
<tr>
<td>PEXPCClipMode</td>
<td>PEXNone</td>
</tr>
<tr>
<td>PEXPCDistinguishFlag</td>
<td>False</td>
</tr>
<tr>
<td>PEXPCPatternSize</td>
<td>&lt;1.0, 1.0&gt;</td>
</tr>
<tr>
<td>PEXPCPatternRefPoint</td>
<td>&lt;0.0, 0.0, 0.0&gt;</td>
</tr>
<tr>
<td>PEXPCPatternRefVec1</td>
<td>&lt;1.0, 0.0, 0.0&gt;</td>
</tr>
<tr>
<td>PEXPCPatternRefVec2</td>
<td>&lt;0.0, 1.0, 0.0&gt;</td>
</tr>
<tr>
<td>PEXPCInteriorBundleIndex</td>
<td>1</td>
</tr>
<tr>
<td>PEXPCSurfaceEdgeFlag</td>
<td>PEXOff</td>
</tr>
<tr>
<td>PEXPCSurfaceEdgeType</td>
<td>PEXSurfaceEdgeSolid</td>
</tr>
<tr>
<td>PEXPCSurfaceEdgeWidth</td>
<td>1.0</td>
</tr>
<tr>
<td>PEXPCSurfaceEdgeColor</td>
<td>{PEXColorTypeIndexed, 1}</td>
</tr>
<tr>
<td>PEXPCEdgeBundleIndex</td>
<td>1</td>
</tr>
<tr>
<td>PEXPLocalTransform</td>
<td>Identity matrix</td>
</tr>
<tr>
<td>PEXPCGlobalTransform</td>
<td>Identity matrix</td>
</tr>
<tr>
<td>PEXPCModelClip</td>
<td>PEXNoClip</td>
</tr>
<tr>
<td>PEXPCModelClipVolume</td>
<td>NULL</td>
</tr>
<tr>
<td>PEXPCViewIndex</td>
<td>0</td>
</tr>
<tr>
<td>PEXPLightState</td>
<td>NULL</td>
</tr>
<tr>
<td>PEXPCDepthCueIndex</td>
<td>0</td>
</tr>
<tr>
<td>PEXPCASFValues</td>
<td>PEXIndividual</td>
</tr>
<tr>
<td>PEXPCPickID</td>
<td>0</td>
</tr>
<tr>
<td>PEXPCHLHSRIdentifier</td>
<td>0</td>
</tr>
<tr>
<td>PEXPCNameSet</td>
<td>NULL</td>
</tr>
<tr>
<td>PEXPCColorApproxIndex</td>
<td>0</td>
</tr>
<tr>
<td>PEXPCRenderingColorModel</td>
<td>0</td>
</tr>
<tr>
<td>PEXPCFarSurfCharacteristics</td>
<td>{1, NULL}</td>
</tr>
</tbody>
</table>

DATA STRUCTURES

typedef XID PEXPipelineContext;

See also PEXlib.h.
### ERRORS

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BadAlloc</td>
<td>The server failed to allocate the resource.</td>
</tr>
<tr>
<td>BadPEXColorType</td>
<td>The specified color type is invalid or unsupported.</td>
</tr>
<tr>
<td>BadPEXNameSet</td>
<td>A specified name set resource identifier is invalid.</td>
</tr>
<tr>
<td>BadValue</td>
<td>A specified value is out of range, or an invalid bit is set in the value mask.</td>
</tr>
</tbody>
</table>

### SEE ALSO

PEXFreePipelineContext(3)
NAME    PEXCreateRenderer – Create Renderer

SYNTAX    PEXRenderer PEXCreateRenderer(Display *display, Drawable drawable, unsigned long value_mask, PEXRendererAttributes *values)

PARAMETERS

display  A pointer to a display structure returned by a successful XOpenDisplay call.
drawable  The resource identifier of a drawable.
value_mask  A mask indicating the attribute values specified.
values  A pointer to values used to override default values in the new renderer resource.

RETURNS  The resource identifier of the newly-created renderer resource.

DESCRIPTION  This function creates a renderer and returns the resource identifier of the newly-created renderer resource. The value mask indicates which attribute values are specified. Attribute values not specified will be initialized to the default values shown in the table below.

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEXRAPipelineContext</td>
<td>NULL</td>
</tr>
<tr>
<td>PEXRAMarkerBundle</td>
<td>NULL</td>
</tr>
<tr>
<td>PEXRATextBundle</td>
<td>NULL</td>
</tr>
<tr>
<td>PEXRALineBundle</td>
<td>NULL</td>
</tr>
<tr>
<td>PEXRAInteriorBundle</td>
<td>NULL</td>
</tr>
<tr>
<td>PEXRAEdgeBundle</td>
<td>NULL</td>
</tr>
<tr>
<td>PEXRAViewTable</td>
<td>NULL</td>
</tr>
<tr>
<td>PEXRAColorTable</td>
<td>NULL</td>
</tr>
<tr>
<td>PEXRADepthCueTable</td>
<td>NULL</td>
</tr>
<tr>
<td>PEXRALightTable</td>
<td>NULL</td>
</tr>
<tr>
<td>PEXRAColorApproxTable</td>
<td>NULL</td>
</tr>
<tr>
<td>PEXRAPatternTable</td>
<td>NULL</td>
</tr>
<tr>
<td>PEXRATextFontTable</td>
<td>NULL</td>
</tr>
<tr>
<td>PEXRAHighlightIncl</td>
<td>NULL</td>
</tr>
<tr>
<td>PEXRAHighlightExcl</td>
<td>NULL</td>
</tr>
<tr>
<td>PEXRAInvisibilityIncl</td>
<td>NULL</td>
</tr>
<tr>
<td>PEXRAInvisibilityExcl</td>
<td>NULL</td>
</tr>
<tr>
<td>PEXRAHLHSRMode</td>
<td>PEXHLHSROff</td>
</tr>
<tr>
<td>PEXRANPSubVolume</td>
<td>((0.0, 0.0, 0.0), (1.0, 1.0, 1.0))</td>
</tr>
<tr>
<td>PEXRAViewport</td>
<td>{imp.dep., imp.dep., True}</td>
</tr>
<tr>
<td>PEXRAClipList</td>
<td>NULL</td>
</tr>
<tr>
<td>PEXRAPickIncl</td>
<td>NULL</td>
</tr>
<tr>
<td>PEXRAPickExcl</td>
<td>NULL</td>
</tr>
<tr>
<td>PEXRAPickStartPath</td>
<td>NULL</td>
</tr>
<tr>
<td>PEXRABackgroundColor</td>
<td>{PEXColortypeIndexed, 0}</td>
</tr>
</tbody>
</table>
The renderer resource may only be used in conjunction with drawables that have the same root and depth as specified drawable.

Note that the renderer's current path and renderer state attributes cannot be set explicitly using this function.

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEXRAClearImage</td>
<td>False</td>
</tr>
<tr>
<td>PEXRAClearZ</td>
<td>True</td>
</tr>
<tr>
<td>PEXRAEchoMode</td>
<td>PEXNoEcho</td>
</tr>
</tbody>
</table>

typedef XID PEXRenderer;

See also PEXlib.h.

ERRORS

BadAlloc
The server failed to allocate the resource.

BadDrawable
The specified drawable resource identifier is invalid.

BadMatch
The specified drawable is unsupported, or the specified lookup table resource was not created with a drawable compatible with the specified drawable.

BadPEXLookupTable
The specified lookup table resource identifier is invalid.

BadPEXNameSet
The specified name set resource identifier is invalid.

BadPEXPipelineContext
The specified pipeline context resource identifier is invalid.

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadValue
A specified value is out of range, or an invalid bit is set in the value mask.

SEE ALSO
PEXFreeRenderer(3)
NAME
PEXCreateSearchContext – Create Search Context

SYNTAX
PEXSearchContext PEXCreateSearchContext(Display *display, unsigned long value_mask,
PEXSCAttributes *values)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay
call.
value_mask A mask indicating the attributes specified.
values A pointer to values used to override default values in the new search
context resource.

RETURNS
The resource identifier of the newly-created search context resource.

DESCRIPTION
This function creates a search context and returns the resource identifier of the created
search context resource. The value mask indicates which values are specified to override
the default values.

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEXSCPosition</td>
<td>&lt;0.0, 0.0, 0.0&gt;</td>
</tr>
<tr>
<td>PEXSCDistance</td>
<td>0.0</td>
</tr>
<tr>
<td>PEXSCCeiling</td>
<td>1</td>
</tr>
<tr>
<td>PEXSCModelClipFlag</td>
<td>False</td>
</tr>
<tr>
<td>PEXSCStartPath</td>
<td>NULL</td>
</tr>
<tr>
<td>PEXSCNormalList</td>
<td>NULL</td>
</tr>
<tr>
<td>PEXSCInvertedList</td>
<td>NULL</td>
</tr>
</tbody>
</table>

DATA STRUCTURES
typedef XID PEXRenderer;
typedef XID PEXSearchContext;
typedef struct {
    PEXCoord position;
    float distance;
    unsigned short ceiling;
    Bool model_clip_flag;
    PEXStructurePath start_path;
    PEXListOfNameSetPair normal;
    PEXListOfNameSetPair inverted;
} PEXSCAttributes;
typedef struct {
    float x;
    float y;
    float z;
}

modified November 1995
typedef struct {
    unsigned long count; /* number of elements */
    PEXElementRef *elements;
} PEXCoord;

typedef struct {
    PEXStructure structure;
    unsigned long offset;
} PEXElementRef;

typedef XID PEXStructure;

typedef struct {
    unsigned short count; /* number of pairs */
    PEXNameSetPair *pairs;
} PEXListOfNameSetPair;

typedef struct {
    PEXNameSet inclusion;
    PEXNameSet exclusion;
} PEXNameSetPair;

typedef XID PEXNameSet;

ERRORS

BadAlloc
The server failed to allocate the resource.

BadPEXNameSet
The specified name set resource identifier is invalid.

BadPEXPath
The specified path is invalid.

BadValue
A specified value is out of range, or an invalid bit is set in the value mask.

SEE ALSO
PEXFreeSearchContext(3)
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXCreateStructure – Create Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td>PEXStructure PEXCreateStructure(Display *display)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>display A pointer to a display structure returned by a successful XOpenDisplay call.</td>
</tr>
<tr>
<td>RETURNS</td>
<td>The resource identifier of the newly-created structure resource.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>This function creates a structure resource and returns the resource identifier of the created structure. The returned identifier is used to refer to the created structure resource.</td>
</tr>
<tr>
<td>DATA STRUCTURES</td>
<td>typedef XID PEXStructure;</td>
</tr>
<tr>
<td>ERRORS</td>
<td>BadAlloc The server failed to allocate the resource.</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>PEXDestroyStructures(3)</td>
</tr>
</tbody>
</table>
NAME
PEXCreateWorkstation – Create Workstation

SYNTAX
PEXWorkstation PEXCreateWorkstation(Display *display, Drawable drawable, PEXLookupTable line_bundle, PEXLookupTable marker_bundle, PEXLookupTable text_bundle, PEXLookupTable interior_bundle, PEXLookupTable edge_bundle, PEXLookupTable color_table, PEXLookupTable pattern_table, PEXLookupTable font_table, PEXLookupTable depth_cue_table, PEXLookupTable light_table, PEXLookupTable color_approx_table, PEXNameSet highlight_incl, PEXNameSet highlight_excl, PEXNameSet invisibility_incl, PEXNameSet invisibility_excl, int buffer_mode)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
drawable The resource identifier of a drawable.
line_bundle The resource identifier of the line bundle lookup table.
marker_bundle The resource identifier of the marker bundle lookup table.
text_bundle The resource identifier of the text bundle lookup table.
interior_bundle The resource identifier of the interior bundle lookup table.
edge_bundle The resource identifier of the edge bundle lookup table.
color_table The resource identifier of the color lookup table.
pattern_table The resource identifier of the pattern lookup table.
font_table The resource identifier of the text font lookup table.
depth_cue_table The resource identifier of the depth cue lookup table.
light_table The resource identifier of the light lookup table.
color_approx_table The resource identifier of the color approximation lookup table.
highlight_incl The name set used as the highlight inclusion set.
highlight_excl The name set used as the highlight exclusion set.
 invisibility_incl The name set used as the invisibility inclusion set.
invisibility_excl The name set used as the invisibility exclusion set.
buffer_mode The workstation buffering mode (PEXSingleBuffer or PEXDoubleBuffer).

RETURNS
The resource identifier of the newly-created workstation.

DESCRIPTION
This function creates a PHIGS workstation resource and returns its resource identifier. The drawable specified is associated with the newly-created workstation resource. The named tables and name sets are also bound to the workstation resource for use during rendering. A view table that supports current and requested view table entries is allocated for the workstation automatically at creation time. The requests

modified November 1995
PEXSetWorkstationViewRep(3) and PEXGetWorkstationViewRep(3) are used to modify and inquire the PHIGS workstation view table.

If the workstation is to operate in double-buffered mode and double-buffering is supported (see PEXGetImpDepConstants(3)), an additional image buffer will be created for the drawable in an implementation-dependent fashion. If the specified drawable is a pixmap, no double-buffering will be performed.

**DATA STRUCTURES**

typedef XID PEXWorkstation;
typedef XID PEXLookupTable;
typedef XID PEXNameSet;

**ERRORS**

BadAlloc

The server failed to allocate the resource, or the server failed to allocate resources needed for double-buffering.

BadDrawable

The specified drawable resource identifier is invalid.

BadMatch

The specified drawable is unsupported, or a specified lookup table resource was not created with a compatible drawable.

BadPEXLookupTable

The specified lookup table resource identifier is invalid.

BadPEXNameSet

The specified name set resource identifier is invalid.

BadValue

The specified buffer mode is invalid.

**SEE ALSO**

PEXFreeWorkstation(3)
PEXGetImpDepConstants(3)
NAME
PEXDecodeOCs – Decode Output Commands

SYNTAX
PEXOCData *PEXDecodeOCs(int float_format, unsigned long oc_count, unsigned long length, char *encoded_ocs)

PARAMETERS
float_format The floating point format of the encoded output commands ( PEXIEEE_754_32, PEXDEC_F_Floating, PEXIEEE_754_64, PEXDEC_D_Floating ).

oc_count The number of output commands represented in the encoded output commands.

length The length, in bytes, of the encoded output commands.

encoded_ocs A pointer to the encoded output commands.

RETURNS
A pointer to the decoded output commands; a null pointer if unsuccessful or if zero output commands specified.

DESCRIPTION
This function has no visible effect. Encoded output commands are passed in and the data typically passed as parameters to output attribute or primitive functions is returned in memory allocated by PEXlib. PEXFreeOCData(3) should be called to deallocate the memory.

A null pointer will be returned if the specified floating point format is not supported. Any text or annotation text primitives are returned as encoded text or encoded annotation text.

PEXCountOCs(3) can be used to determine the number of output commands in the encoded list if that information is not already available.

ERRORS
None

SEE ALSO
PEXEncodeOCs(3) PEXCountOCs(3)
NAME
PEXDeleteBetweenLabels – Delete Elements Between Labels

SYNTAX
void PEXDeleteBetweenLabels(  
Display *display, PEXStructure structure, long label1, long label2)

PARAMETERS
display  A pointer to a display structure returned by a successful XOpenDisplay 
call.
structure The resource identifier of the structure.
label1 The first label.
label2 The second label.

RETURNS
None

DESCRIPTION
This function deletes a range of elements from the specified structure. Elements between 
the two labels are deleted. The label elements are not deleted. A search for the first label 
is performed starting at the current offset plus one. A search for the second label is per-
formed starting at the element following the first label. After the deletion operation, the 
structure element pointer is set to the pointer position at the first label.

If either of the two labels is not found between the starting point of the search and the 
end of the structure, no deletion occurs and the structure’s element point is left 
unchanged.

DATA
STRUCTURES
typedef XID PEXStructure;

ERRORS
BadPEXLabel
The specified label does not exist.
BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO
PEXCreateStructure(3)  
PEXLabel(3)
NAME
PEXDeleteElements – Delete Elements

SYNTAX
void PEXDeleteElements(Display *display, PEXStructure structure, int whence1, long offset1, int whence2, long offset2)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
structure The resource identifier of the structure.
whence1 A value specifying, with offset1, the first limit of the range of elements to be deleted (PEXBeginning, PEXCurrent, PEXEnd).
offset1 The offset from whence1 denoting the first limit of the range of elements to be deleted.
whence2 A value specifying, with offset2, the second limit of the range of elements to be deleted (PEXBeginning, PEXCurrent, PEXEnd).
offset2 The offset from whence2 denoting the second limit of the range of elements to be deleted.

RETURNS
None

DESCRIPTION
This function deletes a range of elements from the structure specified by structure. If a computed offset is less than zero, it is set to zero before obtaining the element information. If a computed offset is greater than the number of elements in the structure, the offset is set to the offset of the last structure element in the structure. Deleting the null element is effectively a no-op. After the deletion operation, the structure element pointer is set to the element immediately preceding the range of deleted elements.

DATA STRUCTURES
typedef XID PEXStructure;

ERRORS
BadPEXStructure The specified structure resource identifier is invalid.
BadValue The specified value for whence parameter is invalid.

SEE ALSO
PEXCreateStructure(3)
NAME  PEXDeleteTableEntries – Delete Lookup Table Entries

SYNTAX  void PEXDeleteTableEntries(Display *display, PEXLookupTable table, unsigned int start, unsigned int count)

PARAMETERS  
| display | A pointer to a display structure returned by a successful XOpenDisplay call. |
| table   | The resource identifier of the lookup table. |
| start   | The initial table entry to be deleted. |
| count   | The number of entries to be deleted. |

RETURNS  None

DESCRIPTION  This function deletes lookup table entries from the specified lookup table, starting at the specified entry. Entries with index values between the start and (start + count-1), inclusive, are deleted. Attempts to delete undefined entries are ignored.

DATA STRUCTURES  typedef XID PEXLookupTable;

ERRORS  
| BadPEXLookupTable | The specified lookup table resource identifier is invalid, or the table type is unsupported. |
| BadValue         | The sum of start and count is too large, or index 0 is invalid for the specified table type. |

SEE ALSO  PEXGetTableInfo(3)  PEXGetPredefinedEntries(3)  PEXGetDefinedIndices(3)
PEXDeleteToLabel – Delete Elements to Label

void PEXDeleteToLabel(Display *display, PEXStructure structure, int whence, long offset, long label)

PARAMETERS

- **display**: A pointer to a display structure returned by a successful XOpenDisplay call.
- **structure**: The resource identifier of the structure.
- **whence**: A value specifying, with offset, the beginning of the range of elements to be deleted (PEXBeginning, PEXCurrent, PEXEnd).
- **offset**: The offset from whence denoting the beginning of the range of elements to be deleted.
- **label**: The label specifying the end of the range of elements to be deleted.

RETURNS

None

DESCRIPTION

This function deletes a range of elements between a computed offset and a specified label in the specified structure. The computed offset specifies the beginning of the deletion range. The label specifies the end of the deletion range. Elements are deleted from the structure element immediately after the computed offset up to the next occurrence of the label. The label is not deleted. If label is not found, no elements are deleted.

If the computed offset is less than zero, it is set to zero before the deletion occurs. If the computed offset is greater than the number of elements in the structure, the offset is set to the offset of the last structure element. Deleting the zero element is effectively a no-op. After the deletion operation, the structure element pointer is set to the element immediately preceding the range of deleted elements.

DATA STRUCTURES

typedef XID PEXStructure;

ERRORS

- **BadPEXLabel**: The specified label does not exist.
- **BadPEXStructure**: The specified structure resource identifier is invalid.
- **BadValue**: The specified value for whence parameter is invalid.

SEE ALSO

PEXCreateStructure(3)
PEXLabel(3)
NAME
PEXDestroyStructures – Destroy Structures

SYNTAX
void PEXDestroyStructures(Display *display, unsigned long count, PEXStructure *structures)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
count The number of structure resource identifiers.
structures An array of resource identifiers of the structures to be destroyed.

RETURNS
None

DESCRIPTION
This function deletes each structure in the list of structure identifiers, removes all references to it in the server, and frees all memory associated with it.

This function also removes any "execute structure" structure elements that reference a structure in the list and unposts any structure in the list from a PHIGS workstation resource to which it is posted. If a structure has any structure elements removed from it as a result of this call, its element pointer will continue to point at the same structure element. However, if the structure element being pointed at is removed, the element pointer will be positioned at the previous structure element.

Any paths in search contexts or pick measures that contain references to a deleted structure may still be inquired. However, if a path in a search context or pick measure resource which contains a destroyed structure resource identifier is later used in a PEXSearchNetwork(3) or PEXUpdatePickMeasure(3) function, a path error is generated.

DATA STRUCTURES
typedef XID PEXStructure;

ERRORS
BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO
PEXCreateStructure(3)
PEXGetStructuresInNetwork(3)
PEXGetAncestors(3)
PEXGetDescendants(3)
PEXChangeStructureRefs(3)
NAME
PEXElementSearch – Element Search

SYNTAX
Status PEXElementSearch(Display *display, PEXStructure structure, int whence, long offset, int direction, unsigned long incl_count, unsigned short *incl_list, unsigned long excl_count, unsigned short *excl_list, unsigned long *elem_offset_return)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
structure The resource identifier of the structure.
whence A value specifying, with offset, the offset at which the search is to begin (PEXBeginning, PXEndCurrent, PXEndEnd).
offset The offset from whence at which the search is to begin.
direction The direction of the search (PEXForward or PXBackward).
incl_count The number of values in inclusion array.
incl_list An array of short integers specifying structure elements to be searched for.
excl_count The number of values in exclusion array.
excl_list An array of short integers specifying structure elements not to be searched for.
elem_offset_return Returns the offset of the element located by the search.

RETURNS
Zero if unsuccessful, non-zero otherwise. The non-zero value will be either PEXFound or PXENotFound depending upon the result of the search.

DESCRIPTION
This function searches for the first occurrence of the specified element type in the specified structure. The search always includes the starting element.

If the computed offset is less than zero, it is set to zero before the search is performed. If the computed offset is greater than the number of elements in the structure, it is set to the offset of the last structure element in the structure.

An element is selected if its element type is contained in inclusion list and is not contained in exclusion list. An element type of PEXOCAll causes all element types to match.

If a structure element type is in both the inclusion and exclusion list, it is excluded.

The search terminates if a match is found or if the limits of the structure are reached. The search progresses from the start point in the specified direction (PEXForward or PXBackward). This is a non-descending search; that is, the search does not include any structures referenced by “execute structure” elements. If the search finds a match, a return status of PEXFound and the offset of the matching element is returned. If the search is unsuccessful, a return status of PXENotFound is returned.

modified November 1995
The element pointer position of *structure* is not changed.

**DATA STRUCTURES**  
typedef XID PEXStructure;

**ERRORS**  
BadPEXStructure  
The specified structure resource identifier is invalid.
BadValue  
The specified value for *whence* or *direction* parameter is invalid.

**SEE ALSO**  
PEXCreateStructure(3)
NAME
PEXEncodeOCs – Encode Output Commands

SYNTAX
char *PEXEncodeOCs(int float_format, unsigned long oc_count, PEXOCData *oc_data,
unsigned long *length_return)

PARAMETERS
float_format    The floating point format of the encoded output commands (PEX-
IEEE_754_32, PEXDEC_F_Floating, PEXIEEE_754_64,
PEXDEC_D_Floating).
oc_count       The number of output commands to be encoded.
oc_data        An array of the output command data.
length_return  Returns the length, in bytes, of the encoded output commands.

RETURNS
A pointer to the encoded output commands; a null pointer if unsuccessful or if zero output
commands specified.

DESCRIPTION
This function has no visible effect. The data typically passed as parameters to output
attribute or primitive functions is passed in and encoded into protocol formatted output
commands. The encoded data is returned in memory allocated by PEXlib. PEXFreeOC-
Data(3) should be called to deallocate the memory.
A null pointer will be returned if the specified floating point format is not supported.
Any text or annotation text primitives must be specified as encoded text or encoded
annotation text.

ERRORS
None

SEE ALSO
PEXDecodeOCs(3)
NAME
PEXEncodedAnnoText – Encoded 3D Annotation Text Primitive

SYNTAX
void PEXEncodedAnnoText(Display *display, XID resource_id, PEXOCRequestType
  req_type, PEXCoord *origin, PEXCoord *offset, unsigned int count, PEXEncoded
  TextData *encoded_text)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay
  call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore,
  PEXOCRenderSingle or PEXOCStoreSingle).
origin The origin of the text string.
offset The relative position of the text string from the origin.
count The number of encoded text strings.
encoded_text An array of encoded text strings.

RETURNS
None

DESCRIPTION
This function creates an annotation text output primitive. This function is similar to
PEXAnnotationText(3), except that multiple encoded text strings are specified. Each text string in the encoded text array has a character set, a character set width, an encoding state, and a list of characters.

The character set is an index into the current font group. Font groups contain individual fonts which are numbered starting at one; a value of three selects the third font in the font group currently being used. If a character set is not available in the current font group then the entire string will be rendered using the default font group. If a character set value is not available in the default font group then that portion of the string will be rendered in an implementation-dependent manner. The character set width indicates the width or size of characters in the strings. Valid values for character set width are PEXCSByte, PEXCSShort and PEXCSLong. The encoding state is provided for use by the application and is not interpreted by the PEX server.

All other aspects of this primitive are the same as PEXAnnotationText(3).

DATA STRUCTURES
typedef struct {
  unsigned short character_set;
  unsigned char character_set_width; /* PEXCSByte, PEXCSShort, PEXCSLong */
  unsigned char encoding_state;
  unsigned char reserved;
  unsigned short length;
  char *ch;
} PEXEncodedTextData;

modified November 1995
### ERRORS

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BadPEXRenderer</td>
<td>The specified renderer resource identifier is invalid.</td>
</tr>
<tr>
<td>BadPEXStructure</td>
<td>The specified structure resource identifier is invalid.</td>
</tr>
</tbody>
</table>

### SEE ALSO

- PEXSetFontFontIndex(3)
- PEXSetFontPrecision(3)
- PEXSetCharExpansion(3)
- PEXSetCharSpacing(3)
- PEXSetFontColorIndex(3)
- PEXSetFontColor(3)
- PEXSetATextHeight(3)
- PEXSetATextUpVector(3)
- PEXSetATextPath(3)
- PEXSetATextAlignment(3)
- PEXSetATextStyle(3)
- PEXSetTextBundleIndex(3)
NAME
PEXEncodedAnnoText2D – Encoded 2D Annotation Text Primitive

SYNTAX
void PEXEncodedAnnoText2D(Display *display, XID resource_id, PEXOCRequestType
req_type, PEXCoord2D *origin, PEXCoord2D *offset, unsigned int count, PEXEn-
codedTextData *encoded_text)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay
call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOC-
Store, PEXOCRenderSingle or PEXOCStoreSingle).
origin The origin of the text string.
offset The relative position of the text string from the origin.
count The number of encoded text strings.
encoded_text An array of encoded text strings.

RETURNS None

DESCRIPTION
This function creates a 2D annotation text output primitive.
This function is similar to PEXAnnotationText2D(3), except that multiple encoded text
strings are specified. Each text string in the encoded text array has a character set, a char-
acter set width, an encoding state, and a list of characters.
The character set is an index into the current font group. Font groups contain individual
fonts which are numbered starting at one; a value of three selects the third font in the font
group currently being used. If a character set is not available in the current font group
then the entire string will be rendered using the default font group. If a character set
value is not available in the default font group then that portion of the string will be ren-
dered in an implementation-dependent manner. The character set width indicates the
width or size of characters in the strings. Valid values for character set width are
PEXCSByte, PEXCSShort and PEXCSLong. The encoding state is provided for use by the
application and is not interpreted by the PEX server.
All other aspects of this primitive are the same as PEXAnnotationText2D(3).

DATA
typedef struct {
    unsigned short character_set;
    unsigned char character_set_width; /* PEXCSByte, PEXCSShort,
    unsigned char encoding_state;
    unsigned short reserved;
    unsigned short length;
    char *ch;
} PEXEncodedTextData;
See also **PEXlib.h**.

**ERRORS**

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BadPEXRenderer</strong></td>
<td>The specified renderer resource identifier is invalid.</td>
</tr>
<tr>
<td><strong>BadPEXStructure</strong></td>
<td>The specified structure resource identifier is invalid.</td>
</tr>
</tbody>
</table>

**SEE ALSO**

- PEXSetFontIndex(3)
- PEXSetTextPrecision(3)
- PEXSetCharExpansion(3)
- PEXSetCharSpacing(3)
- PEXSetTextColorIndex(3)
- PEXSetTextColor(3)
- PEXSetATextHeight(3)
- PEXSetATextUpVector(3)
- PEXSetATextPath(3)
- PEXSetATextAlignment(3)
- PEXSetATextStyle(3)
- PEXSetTextBundleIndex(3)
NAME
PEXEncodedText – Encoded 3D Text Primitive

SYNTAX
void PEXEncodedText(Display *display, XID resource_id, PEXOCRequestType req_type,
PEXCoord *origin, PEXVector *vector1, PEXVector *vector2, unsigned int count,
PEXEncodedTextData *encoded_text)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
origin The origin of the text string.
vector1 A vector defining the positive x-direction of the text local coordinate system.
vector2 A vector defining the positive y-direction of the text local coordinate system.
count The number of encoded text strings.
encoded_text An array of encoded text strings.

RETURNS None

DESCRIPTION
This function creates a text output primitive. This function is similar to PEXText(3), except that multiple encoded text strings are specified. Each text string in encoded text array has a character set, a character set width, an encoding state, and a list of characters.

The character set is an index into the current font group. Font groups contain individual fonts which are numbered starting at one; a value of three selects the third font in the font group currently being used. If a character set is not available in the current font group then the entire string will be rendered using the default font group. If a character set value is not available in the default font group then that portion of the string will be rendered in an implementation-dependent manner. The character set width indicates the width or size of characters in the strings. Valid values for character set width are PEXCSByte, PEXCSShort and PEXCSLong. The encoding state is provided for use by the application and is not interpreted by the PEX server. All other aspects of this primitive are the same as PEXText(3).

DATA STRUCTURES
typedef struct {
  unsigned short character_set;
  unsigned char character_set_width; /* PEXCSByte, PEXCSShort, PEXCSLong */
  unsigned char encoding_state;
  unsigned short reserved;
} PEXEncodedTextData;

modified November 1995 69
unsigned short length;
char *ch;
} PEXEncodedTextData;

See also PEXlib.h.

ERRORS

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO

PEXSetFontFontIndex(3)
PEXSetFontPrecision(3)
PEXSetFontExpansion(3)
PEXSetFontSpacing(3)
PEXSetFontColorIndex(3)
PEXSetFontColor(3)
PEXSetFontHeight(3)
PEXSetFontUpVector(3)
PEXSetFontPath(3)
PEXSetFontAlignment(3)
PEXSetFontBundleIndex(3)
NAME
PEXEncodedText2D – Encoded 2D Text Primitive

SYNTAX
void PEXEncodedText2D(Display *display, XID resource_id, PEXOCRequestType
req_type, PEXCoord2D *origin, unsigned int count, PEXEncodedTextData
*encoded_text)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay
call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOC-
Store, PEXOCRenderSingle or PEXOCStoreSingle).
origin The origin of the text string.
count The number of encoded text strings.
encoded_text An array of encoded text strings.

RETURNS None

DESCRIPTION
This function creates a 2D text output primitive.
This function is similar to PEXTxt2D(3), except that multiple encoded text strings are
specified. Each text string in encoded text array has a character set, a character set width,
an encoding state, and a list of characters.

The character set is an index into the current font group. Font groups contain individual
fonts which are numbered starting at one; a value of three selects the third font in the font
group currently being used. If a character set is not available in the current font group
then the entire string will be rendered using the default font group. If a character set
value is not available in the default font group then that portion of the string will be ren-
dered in an implementation-dependent manner. The character set width indicates the
width or size of characters in the strings. Valid values for character set width are
PEXCSByte, PEXCSShort and PEXCSLong. The encoding state is provided for use by the
application and is not interpreted by the PEX server.

All other aspects of this primitive are the same as PEXTxt2D(3).

DATA
STRUCTURES
typedef struct {
    unsigned short character_set;
    unsigned char character_set_width; /* PEXCSByte, PEXCSShort,
                                          PEXCSLong */
    unsigned char encoding_state;
    unsigned short reserved;
    unsigned short length;
    char *ch;
} PEXEncodedTextData;

See also PEXlib.h.

modified November 1995
## ERRORS

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BadPEXRenderer</td>
<td>The specified renderer resource identifier is invalid.</td>
</tr>
<tr>
<td>BadPEXStructure</td>
<td>The specified structure resource identifier is invalid.</td>
</tr>
</tbody>
</table>

## SEE ALSO

- PEXSetFontIndex(3)
- PEXSetTextPrecision(3)
- PEXSetCharExpansion(3)
- PEXSetCharSpacing(3)
- PEXSetTextColorIndex(3)
- PEXSetTextColor(3)
- PEXSetCharHeight(3)
- PEXSetCharUpVector(3)
- PEXSetTextPath(3)
- PEXSetTextAlignment(3)
- PEXSetTextBundleIndex(3)
### NAME
PEXEndPickAll – End Pick All

### SYNTAX
```
PEXPickPath *PEXEndPickAll(Display *display, PEXRenderer renderer, int *status_return,
                          int *more_return, unsigned long *count_return)
```

### PARAMETERS
- **display**: A pointer to a display structure returned by a successful `XOpenDisplay` call.
- **renderer**: The resource identifier of the renderer.
- **status_return**: Returns the status of the pick operation.
- **more_return**: Returns the status of remaining picks.
- **count_return**: Returns the number of pick paths.

### RETURNS
An array of pick paths; a null pointer if unsuccessful or no pick (see also `status_return`).

### DESCRIPTION
This function terminates an immediate-mode pick, returns the hierarchical pick paths of any hit primitives, and sets the renderer state to `PEXIdle`.

If one or more primitives were picked, a pick status of `PEXPick` is returned along with the pick paths. The hierarchical pick path is equivalent to the renderer’s current path at the time the picked primitive was processed. If no primitives were picked, the returned pick status is `PEXNoPick`, and the returned pick paths is a null pointer. If the renderer’s drawable was destroyed or resized during the pick operation, the returned pick status is `PEXAbortedPick` and the returned pick paths is a null pointer.

If all hits were recorded then `PEXNoMoreHits` is returned and the renderer’s pick start path will be empty. If the maximum number of hits was reached and additional hits were detected, then `PEXMoreHits` is returned and the renderer’s pick start path will be set to the last recorded hit primitive. If, after reaching the maximum number of hits, subsequent output commands were ignored, then `PEXMayBeMoreHits` is returned and the renderer’s pick start path is set to the last element processed by the `renderer` before it started ignoring primitives.

If the renderer state is `PEXIdle` when this function is called, (i.e., no picking is in progress or the rendering was aborted due to a resize), the function is ignored and no error is generated. If the renderer state is currently `PEXRendering` or if the pick operation in progress is a pick one, then a `BadPEXRendererState` error is sent.

PEXlib allocates memory for the return value. `PEXFreePickPaths(3)` should be called to deallocate the memory.

### DATA STRUCTURES
```c
typedef XID PEXRenderer;

typedef struct {
    unsigned long count; /* number of elements */
    PEXPickElementRef *elements;
} PEXPickPath;
```

modified November 1995
typedef struct {
    PEXStructure sid;
    unsigned long offset;
    unsigned long pick_id;
} PEXPickElementRef;

typedef XID PEXStructure;

ERRORS

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXRendererState
The specified renderer was in an invalid state.

SEE ALSO
PEXBeginPickAll(3)
PEXPickAll(3)
PEXFreePickPaths(3)
NAME
PEXEndPickOne – End Pick One

SYNTAX
PEXPickPath *PEXEndPickOne(Display *display, PEXRenderer renderer, int *
status_return, int *undetectable_return)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay
  call.
renderer The resource identifier of the renderer.
status_return Returns the status of the pick operation.
undetectable_return Returns True or False indicating whether another pick better satisfied the
  pick criteria with the exception that it did not pass the pick filter test.

RETURNS
A pointer to the pick path; a null pointer if unsuccessful or no pick (see also status_return).

DESCRIPTION
This function terminates an immediate-mode pick, returns the hierarchical pick path to
the closest or last hit primitive, and sets the renderer state to PEXIdle.
If a primitive was picked, the returned pick status is PEXPick. If no primitive was picked,
the returned pick status is PEXNoPick, and the returned pick path is a null pointer. If the
renderer’s drawable was destroyed or resized during the pick operation, the returned
pick status is PEXAbortedPick and the returned pick path is a null pointer.
If there was a primitive which more closely satisfied the pick criteria, but did not pass the
pick filter test, then the undetectable pick return status will be True. Otherwise, it will be
False.
If the renderer state is currently PEXIdle when this function is called, (i.e., no picking is
in progress or the rendering was aborted due to a resize), the function is ignored and no
error is generated. If the renderer state is currently PEXRendering or if the pick opera-
tion in progress is a pick all, then a BadPEXRendererState error is sent.
PExlib allocates memory for the return value. PEXFreePickPaths(3) should be called to
deallocate the memory.

DATA
typedef XID PEXRenderer;

STRUCTURES
typedef struct {
  unsigned long count;       /* number of elements */
  PEXPickElementRef *elements;
} PEXPickPath;

modified November 1995
typedef struct {
    PEXStructure sid;
    unsigned long offset;
    unsigned long pick_id;
} PEXPickElementRef;

typedef XID PEXStructure;

**ERRORS**

**BadPEXRenderer**
The specified renderer resource identifier is invalid.

**BadPEXRendererState**
The specified renderer was in an invalid state.

**SEE ALSO**

PEXBeginPickOne(3)
PEXPickOne(3)
NAME
PEXEndRendering - End Rendering

SYNTAX
void PEXEndRendering(Display *display, PEXRenderer renderer, int flush)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
renderer The resource identifier of the renderer.
flush True or False — specifying whether any pending output for renderer is to be rendered onto its associated drawable.

RETURNS
None

DESCRIPTION
This function terminates rendering on the specified renderer resource. If flush is True, pending output is rendered onto its associated drawable. If flush is False, pending output is discarded. In either case, the renderer state is set to PEXIdle. If the renderer state is PEXIdle, the request is ignored. If the renderer state is PEXPicking, then an error is generated.

ERRORS
BadPEXRenderer
The specified renderer resource identifier is invalid.
BadPEXRendererState
The specified renderer was in an invalid state.

SEE ALSO
PEXCreateRenderer(3)
PEXBeginRendering(3)

modified November 1995 77
**NAME**  
PEXEndStructure - Restore Rendering Pipeline State

**SYNTAX**  
void PEXEndStructure(Display *display, PEXRenderer renderer)

**PARAMETERS**  
- `display`  
  A pointer to a display structure returned by a successful `XOpenDisplay` call.

- `renderer`  
  The resource identifier of the renderer.

**RETURNS**  
None

**DESCRIPTION**  
Application programs can use this function to simulate the side effects of the return from an "execute structure" output command. This request restores the last-saved rendering pipeline attributes in `renderer`. This request also removes the last element reference in the renderer's current path. Subsequent output commands cause the element offset of the element reference at the end of the list to be incremented.

**ERRORS**  
- **BadPEXRenderer**  
The specified renderer resource identifier is invalid.

- **BadPEXRendererState**  
The specified renderer was in an invalid state.

**SEE ALSO**  
PEXBeginStructure(3)  
PEXExecuteStructure(3)  
PEXCreateRenderer(3)
NAME PEXEscape - PEX escape

SYNTAX

void PEXEscape(Display *display, unsigned long escape_id, int length, char *escape_data)

PARAMETERS

display A pointer to a display structure returned by a successful XOpenDisplay call.
escape_id The escape identifier.
length The length, in bytes, of data for the escape request.
escape_data A pointer to data for the escape request.

RETURNS None

DESCRIPTION PEXEscape has an implementation-dependent effect. It is provided as a way for implementation-specific functionality to be accessed. The complete interface and behavior for each specified escape identifier should be specified by the individual PEX server implementation.

The table below lists the supported escape identifiers.

<table>
<thead>
<tr>
<th>Escape Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEXSunEscIdDefineMarkerType</td>
<td>Define marker type</td>
</tr>
<tr>
<td>PEXSunEscIdChangeExtRendAttr</td>
<td>Change extended renderer attributes</td>
</tr>
<tr>
<td>PEXSunEscIdFlushRenderer</td>
<td>Flush renderer</td>
</tr>
<tr>
<td>ES_ESCAPE_DBLBUFFER</td>
<td>Set double buffer mode</td>
</tr>
<tr>
<td>ES_ESCAPE_SWAPBUFFER</td>
<td>Swap buffer</td>
</tr>
<tr>
<td>HP_ESCAPE_DFRONT</td>
<td>Set Draw to front or back buffer</td>
</tr>
</tbody>
</table>

ERRORS BadValue

The specified escape identifier is unsupported.

SEE ALSO PEXEscapeWithReply(3), PEXSetEchoColor(3)
NAME
PEXEscapeWithReply - PEX Escape With Reply

SYNTAX
char *PEXEscapeWithReply(Display *display, unsigned long escape_id, int length, char *escape_data, unsigned long *reply_length_return)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
escape_id The escape identifier.
length The length, in bytes, of data for the escape request.
escape_data A pointer to data for the escape request.
reply_length_return Returns the length, in bytes, of the reply data.

RETURNS
A pointer to the escape reply data; a null pointer if unsuccessful.

DESCRIPTION
PEXEscapeWithReply has an implementation-dependent effect. It is similar to PEXEscape(3) except it has return data. It is provided as a way for implementation-specific functionality to be accessed. The complete interface and behavior for each specified escape identifier should be specified by the individual PEX server implementation.

If the specified escape identifier is not supported, a value error is generated. The table below lists the supported escape identifiers.

The reply data is returned in memory allocated by PEXlib. XFree should be called to deallocate the memory.

<table>
<thead>
<tr>
<th>Escape Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEXSunEscIdGetMarkerDescr</td>
<td>Get marker descriptions</td>
</tr>
<tr>
<td>PEXSunEscIdGetExtRendAttr</td>
<td>Get extended renderer attributes</td>
</tr>
<tr>
<td>PEXSunEscIdGetExtRendAttrDyn</td>
<td>Get extended renderer dynamics</td>
</tr>
<tr>
<td>ES_ESCAPE_SWAPBUFFERCONTENT</td>
<td>Inquire update action during buffer swap</td>
</tr>
</tbody>
</table>

ERRORS
Escape-dependent
See documentation provided with the individual PEX server implementation.

BadValue
The specified escape identifier is unsupported.

SEE ALSO
PEXEscape(3)
NAME
PEXEscape-ES_ESCAPE_DBLBUFFER - E & S Double-Buffering Request

SYNTAX
void PEXEscape(Display *display, unsigned long escape_id, int length, char *escape_data)

PARAMETERS
- display: A pointer to a display structure returned by a successful XOpenDisplay call.
- escape_id: The escape identifier.
- length: The length, in bytes, of data for the escape request.
- escape_data: A pointer to data for the escape request.

RETURNS
None

DESCRIPTION
To use the E & S DoubleBufferingRequest escape, the escape_id parameter should be set to
ES_ESCAPE_DBLBUFFER, the escape_data parameter should be set to point to structure type
esEscapeDblBuffer, and the length parameter should be set to the total length of escape_data.
Sending this escape when the RendererState is Rendering will have an effect that is
implementation-dependent.
This escape is not meant to be used for mixing X and PEX graphics. Attempts to do so
when the renderer is in double-buffer mode will produce implementation-dependent
results.

Note: As stated in Chapter 2 and Chapter 3 of the Solaris PEX Implementation
Specification manual, PEX does not interoperate with the MBX extension.

DATA STRUCTURES
typedef struct {
    Drawable drawable;
    unsigned long bufferMode;
} esEscapeDblBuffer;

Values for bufferMode are listed below:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>ES_RENDERER_DBLBUFFER</td>
<td>1</td>
<td>Allocate a back (undisplayed) buffer. Drawing commands directed at drawable will be written into the back buffer. If drawable is already double-buffered, this will have no effect.</td>
</tr>
<tr>
<td>ES_RENDERER_SINGLEBUFFER</td>
<td>0</td>
<td>Deallocate the back buffer. If drawable is already single-buffered, this will have no effect.</td>
</tr>
</tbody>
</table>

ERRORS
- BadValue
The specified vendorId or escapeType are not supported by the server.
- BadValue
Returned if bufferMode is neither ES_RENDERER_SINGLEBUFFER nor
ES_RENDERERS_DBLBUFFER.

BadDrawable
   The specified Drawable resource ID is invalid.

BadAlloc
   The second buffer cannot be allocated.

SEE ALSO  PEXEscape(3)
NAME
PEXEscape-ES_ESCAPE_SWAPBUFFER - E & S Swap Buffer Request

SYNTAX
void PEXEscape(Display *display, unsigned long escape_id, int length, char *escape_data)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
escape_id The escape identifier.
length The length, in bytes, of data for the escape request.
escape_data A pointer to data for the escape request.

RETURNS
None

DESCRIPTION
To use the E & S SwapBufferRequest escape, the escape_id parameter should be set to ES_ESCAPE_SWAPBUFFER the escape_data parameter should be set to point to structure type esEscapeSwapBuffer and the length parameter should be set to the total length of escape_data.

This request swaps buffers on the specified drawable. The undisplayed buffer becomes the displayed buffer and the previously displayed buffer will be in a state described by the value returned by the Swap Buffer Content Escape Request.

Sending this escape when the RendererState is Rendering will have an effect that is implementation dependent. Sending this escape with a drawable that is not double-buffered will have no effect.

DATA STRUCTURES
typedef struct {
    Drawable drawable;
} esEscapeSwapBuffer;

ERRORS
BadDrawable
The specified Drawable resource ID is invalid.

BadValue
The vendorId and the escapeType are not supported by the server.

SEE ALSO
PEXEscape(3)
NAME
PEXEscape-HP_ESCAPE_DFRONT - HP Set Rendering Buffer Escape

SYNTAX
void PEXEscape(Display ∗display, unsigned long escape_id, int length, char ∗escape_data)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
escape_id The escape identifier.
length The length, in bytes, of data for the escape request.
escape_data A pointer to data for the escape request.

RETURNS
None

DESCRIPTION
To use the HP SetDrawBuffer escape, the escape_id parameter should be set to HP_ESCAPE_DFRONT, the escape_data parameter should be set to point to structure type hpEscapeSetRenderingBuffer, and the length parameter should be set to the total length of escape_data.

This escape allows you to specify which buffer to draw to, allowing you, for example, to draw something immediately to the displayed buffer.

DATA STRUCTURES
The data structure for Hewlett-Packard Set Rendering buffer is defined as a data record to follow the standard escape request header data structure.

typedef struct {
    Drawable drawable;
    int render_to_front_buffer;
} hpEscapeSetRenderingBuffer

Possible values for render_to_front_buffer are defined in the header file HPpex.h as follows:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>HP_RENDER_TO_BACK_BUFFER</td>
<td>0</td>
<td>Draw to the undisplayed buffer.</td>
</tr>
<tr>
<td>HP_RENDER_TO_FRONT_BUFFER</td>
<td>1</td>
<td>Draw to the displayed buffer.</td>
</tr>
</tbody>
</table>

ERRORS
BadDrawable
The specified Drawable resource ID is invalid.

BadValue
The value for render_to_front_buffer was incorrect.

SEE ALSO
PEXEscape(3)

modified November 1995
NAME    PEXEscape-PEXSunEscIdChangeExtRendAttr - PEX Change Extended Renderer Attributes

SYNTAX  void PEXEscape(Display *display, unsigned long escape_id, int length, char *escape_data)

PARAMETERS  
  display  A pointer to a display structure returned by a successful XOpenDisplay call.
  escape_id  The escape identifier.
  length  The length, in bytes, of data for the escape request.
  escape_data  A pointer to data for the escape request.

RETURNS  None

DESCRIPTION  To use the Sun ChangeExtendedRendererAttributes escape, the escape_id parameter should be set to PEXSunEscIdChangeExtRendAttr, the escape_data parameter should be set to point to structure type pexSunEscChangeExtRendAttr, and the length parameter should be set to the total length of escape_data.

This escape allows modification of one or more of the extended renderer attributes. The semantics of this escape are the same as those of PEXChangeRenderer(3).

DATA STRUCTURES  
typedef struct {
  pexEnumTypeIndex fpFormat B16;
  CARD16 unused B16;
  XID id B32; /* renderer id */
  CARD32 itemMask;
  /* LISTofVALUE */
} pexSunEscChangeExtRendAttr;

Values for itemMask are defined in SunPEX.h as listed below:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEXRDSunAAliasMode</td>
<td>0x00000001</td>
<td>Anti-aliasing mode attribute</td>
</tr>
<tr>
<td>PEXRDSunTranspMode</td>
<td>0x00000002</td>
<td>Transparency mode attribute</td>
</tr>
<tr>
<td>PEXRDSunStereoMode</td>
<td>0x00000004</td>
<td>Stereo mode attribute</td>
</tr>
<tr>
<td>PEXRDSunSilhouetteEdgeMode</td>
<td>0x00000008</td>
<td>Silhouette edge mode attribute</td>
</tr>
</tbody>
</table>

SEE ALSO  PEXEscape(3)  
PEXChangeRenderer(3)
NAME | PEXEscape-PEXSunEscIdDefineMarkerType - PEX Define Marker Escape

SYNTAX | void PEXEscape(Display *display, unsigned long escape_id, int length, char *escape_data)

PARAMETERS |
| display | A pointer to a display structure returned by a successful XOpenDisplay call.
| escape_id | The escape identifier.
| length | The length, in bytes, of data for the escape request.
| escape_data | A pointer to data for the escape request.

RETURNS | None

DESCRIPTION | To use the Sun DefineMarkerType escape, the escape_id parameter should be set to PEXSunEscIdDefineMarkerType, the escape_data parameter should be set to point to structure type pexSunEscDefineMarkerType, and the length parameter should be set to the total length of escape_data. This escape allows the application to define a marker as a set of unconnected polylines. This set is constructed of point sequences; each point sequence generates connected line segments. The points are specified in marker nominal coordinates: [-1.0, 1.0] in both x and y.

DATA STRUCTURES | typedef struct {
| pexEnumTypeIndex fpFormat B16;
| CARD16 unused B16;
| CARD32 marker_id;
| CARD32 num_lists;
| /* LISTof LISTof COORD2D */
} pexSunEscDefineMarkerType;

ERRORS | BadValue
| The length field of the escape is too small.

SEE ALSO | PEXEscape(3)

86  modified November 1995
NAME
PEXEscape-PEXSunEscIdFlushRenderer - PEX Flush Renderer Escape

SYNTAX
void PEXEscape(Display *display, unsigned long escape_id, int length, char *escape_data)

PARAMETERS
- **display**: A pointer to a display structure returned by a successful XOpenDisplay call.
- **escape_id**: The escape identifier.
- **length**: The length, in bytes, of data for the escape request.
- **escape_data**: A pointer to data for the escape request.

RETURNS
None

DESCRIPTION
To use the Sun FlushRenderer escape, the `escape_id` parameter should be set to PEX-SunEscIdFlushRenderer, the `escape_data` parameter should be set to point to structure type `pexSunEscFlushRendererData`, and the `length` parameter should be set to the total length of `escape_data`.

This escape forces all pending output commands to be either drawn or discarded. If `flush` is `True`, all pending output is rendered onto the drawable associated with the renderer, or if the renderer state is `Picking`, all pending output is processed. If `flush` is `False`, all pending output is discarded. If the renderer state is `Idle`, this escape has no effect.

DATA STRUCTURES
typedef struct {
    CARD32 rdr B32;
    CARD8 flush;
    CARD8 unused[3];
} pexSunEscFlushRendererData;

SEE ALSO
PEXEscape(3)

modified November 1995
**NAME**

PEXEscapeWithReply-ES_ESCAPE_SWAPBUFFERCONTENT - E & S Inquire Swap Buffer Content Request

**SYNTAX**

```c
char *PEXEscapeWithReply(Display *display, unsigned long escape_id, int length, char *escape_data, unsigned long *reply_length_return)
```

**PARAMETERS**

- `display` A pointer to a display structure returned by a successful `XOpenDisplay` call.
- `escape_id` The escape identifier.
- `length` The length, in bytes, of data for the escape request.
- `escape_data` A pointer to data for the escape request.
- `reply_length_return` Returns the length, in bytes, of the reply data.

**RETURNS**

A pointer to the escape reply data; a null pointer if unsuccessful.

**DESCRIPTION**

To use the E & S SwapBufferContentRequest escape, the `escape_id` parameter should be set to `ES_ESCAPE_SWAPBUFFERCONTENT` the `escape_data` parameter should be set to point to structure type `esEscapeSwapBufferContent` and the `length` parameter should be set to the total length of `escape_data`.

This inquiry reports to the client what will be in the back buffer after a swap buffer escape request has occurred. Since this escape always returns the same value for a given `drawable`, it is unnecessary to issue this escape after every swap because the state of the previously displayed buffer remains consistent for `drawable`.

This inquiry returns a pointer to an unsigned long `content`. Possible values returned for `content` are listed below:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>ES_DB_SWAP_CONTENT_UNDEFINED</td>
<td>0</td>
<td>The content of the previously displayed buffer is undefined.</td>
</tr>
<tr>
<td>ES_DB_SWAP_CONTENT_CLEAR_TO_BACKGROUND*</td>
<td>1</td>
<td>The previously displayed buffer is cleared to background after the swap buffer request.</td>
</tr>
<tr>
<td>ES_DB_SWAP_CONTENT_UNCHANGED</td>
<td>2</td>
<td>The previously displayed buffer content is unchanged after the swap buffer request.</td>
</tr>
<tr>
<td>ES_DB_SWAP_CONTENT_FRONTBUFFER</td>
<td>3</td>
<td>The previously displayed buffer has the same content as the now currently displayed buffer after the swap buffer request.</td>
</tr>
</tbody>
</table>

88 modified November 1995
typedef struct {
    Drawable drawable;
} esEscapeSwapBufferContent;

ERRORS
BadDrawable
    The specified Drawable resource ID is invalid.

SEE ALSO
PEXEscapeWithReply(3)
NAME  PEXEscapeWithReply-PEXSunEscIdGetExtRendAttr - PEX Get Extended Renderer Attributes

SYNTAX  char *PEXEscapeWithReply(Display *display, unsigned long escape_id, int length, char *escape_data, unsigned long *reply_length_return)

PARAMETERS  display  A pointer to a display structure returned by a successful XOpenDisplay call.
  escape_id  The escape identifier.
  length  The length, in bytes, of data for the escape request.
  escape_data  A pointer to data for the escape request.
  reply_length_return  Returns the length, in bytes, of the reply data.

RETURNS  A pointer to the escape reply data; a null pointer if unsuccessful.

DESCRIPTION  To use the Sun GetExtendedRendererAttributes escape, the escape_id parameter should be set to PEXSunEscIdGetExtRendAttr, the escape_data parameter should be set to point to structure type pexSunEscGetExtRendAttr, and the length parameter should be set to the total length of escape_data.

This escape allows inquiry of one or more of the extended renderer attributes. The semantics of this escape are the same as those of PEXGetRendererAttributes(3).

This escape returns a pointer to reply data in the following format:

<table>
<thead>
<tr>
<th>unused[20 bytes]</th>
</tr>
</thead>
<tbody>
<tr>
<td>value(1) [4 bytes]</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>value(n) [4 bytes]</td>
</tr>
</tbody>
</table>

where value(i) corresponds to the i-th bit of the renderer attribute set in the itemMask.

DATA STRUCTURES  typedef struct {
  pexEnumTypeIndex fpFormat B16;
  CARD16 unused B16;
  XID id B32;     /* renderer id */
  CARD32 itemMask;
} pexSunEscGetExtRendAttr;

Values for itemMask are defined in SunPEX.h as listed below:
## Symbol Value Explanation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEXRDSunAAliasMode</td>
<td>0x00000001</td>
<td>Anti-aliasing mode attribute</td>
</tr>
<tr>
<td>PEXRDSunTranspMode</td>
<td>0x00000002</td>
<td>Transparency mode attribute</td>
</tr>
<tr>
<td>PEXRDSunStereoMode</td>
<td>0x00000004</td>
<td>Stereo mode attribute</td>
</tr>
<tr>
<td>PEXRDSunSilhouetteEdgeMode</td>
<td>0x00000008</td>
<td>Silhouette edge mode attribute</td>
</tr>
</tbody>
</table>

### SEE ALSO
- PEXEscape(3)
- PEXEscapeWithReply(3)
- PEXGetRendererAttributes(3)
NAME
PEXEscapeWithReply-PEXSunEscIdGetExtRendAttrDyn - PEX Get Extended Renderer Attributes Dynamics

SYNTAX
char *PEXEscapeWithReply(Display *display, unsigned long escape_id, int length, char *escape_data, unsigned long *reply_length_return)

PARAMETERS
- **display**: A pointer to a display structure returned by a successful XOpenDisplay call.
- **escape_id**: The escape identifier.
- **length**: The length, in bytes, of data for the escape request.
- **escape_data**: A pointer to data for the escape request.
- **reply_length_return**: Returns the length, in bytes, of the reply data.

RETURNS
A pointer to the escape reply data; a null pointer if unsuccessful.

DESCRIPTION
To use the Sun GetExtendedRendererAttributesDynamics escape, the escape_id parameter should be set to PEXSunEscIdGetExtRendAttrDyn, the escape_data parameter should be set to point to structure type pexSunEscGetExtRendAttrDyn, and the length parameter should be set to the total length of escape_data.

This escape allows inquiry of the extended renderer dynamics. The semantics of this escape are the same as those of PEXGetRendererDynamics(3). This information is renderer-dependent.

This escape returns a pointer to reply data itemMask, of type unsigned long. Values for itemMask are defined in SunPEX.h as listed below:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEXRDSunAAliasMode</td>
<td>0x00000001</td>
<td>Anti-aliasing mode attribute is dynamic</td>
</tr>
<tr>
<td>PEXRDSunTranspMode</td>
<td>0x00000002</td>
<td>Transparency mode attribute is dynamic</td>
</tr>
<tr>
<td>PEXRDSunStereoMode</td>
<td>0x00000004</td>
<td>Stereo mode attribute is dynamic</td>
</tr>
<tr>
<td>PEXRDSunSilhouetteEdgeMode</td>
<td>0x00000008</td>
<td>Silhouette edge mode attribute is dynamic</td>
</tr>
</tbody>
</table>

DATA STRUCTURES
typedef struct {
  XID id B32;
} pexSunEscGetExtRendAttrDyn;

SEE ALSO
PEXEscape(3)
PEXEscapeWithReply(3)
PEXGetRendererDynamics(3)

92 modified November 1995
NAME
PEXEscapeWithReply-PEXSunEscIdGetMarkerDescr - PEX Get Marker Description

SYNTAX
char *PEXEscapeWithReply(Display *display, unsigned long escape_id, int length, char *escape_data, unsigned long *reply_length_return)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
escape_id The escape identifier.
length The length, in bytes, of data for the escape request.
escape_data A pointer to data for the escape request.
reply_length_return Returns the length, in bytes, of the reply data.

RETURNS
A pointer to the escape reply data; a null pointer if unsuccessful.

DESCRIPTION
To use the Sun GetMarkerDescription escape, the escape_id parameter should be set to PEXSunEscIdGetMarkerDescr, the escape_data parameter should be set to point to structure type pexSunEscGetMarkerDescr, and the length parameter should be set to the total length of escape_data.

This escape allows inquiry of a user-defined marker.

This escape returns a pointer to reply data in the following format:

```
unused[20 bytes]
num_lists [4 bytes]
list(1)
... list(num_lists)
```

Each list contains the following:

```
num_verts [4 bytes]
x(1) [float]
y(1) [float]
... x[num_verts]
y[num_verts]
```

DATA STRUCTURES
typedef struct {
pexEnumTypeIndex fpFormat B16;
CARD16 unused B16;
CARD32 marker_id;
} pexSunEscGetMarkerDescr;
SEE ALSO

PEXEscape(3)
PEXEscapeWithReply(3)
<table>
<thead>
<tr>
<th><strong>NAME</strong></th>
<th>PEXExecuteDeferredActions - Execute Deferred Workstation Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SYNTAX</strong></td>
<td>void PEXExecuteDeferredActions(Display *display, PEXWorkstation workstation)</td>
</tr>
<tr>
<td><strong>PARAMETERS</strong></td>
<td>display A pointer to a display structure returned by a successful XOpenDisplay call.</td>
</tr>
<tr>
<td></td>
<td>workstation The resource identifier of the workstation.</td>
</tr>
<tr>
<td><strong>RETURNS</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>DESCRIPTION</strong></td>
<td>This function causes all deferred actions on the specified workstation to be executed. This will cause all requested attributes to be made current and the corresponding update attributes to be set to PEXNotPending.</td>
</tr>
<tr>
<td><strong>DATA STRUCTURES</strong></td>
<td>typedef XID PEXWorkstation;</td>
</tr>
<tr>
<td><strong>ERRORS</strong></td>
<td>BadPEXWorkstation The specified workstation resource identifier is invalid.</td>
</tr>
<tr>
<td><strong>SEE ALSO</strong></td>
<td>PEXRedrawAllStructures(3)</td>
</tr>
</tbody>
</table>
NAME
PEXExecuteStructure - Execute Structure

SYNTAX
void PEXExecuteStructure(Display *display, XID resource_id, PEXOCRequestType req_type, PEXStructure structure)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
structure The resource identifier of the structure.

RETURNS None

DESCRIPTION
This function creates an execute structure output command which causes the specified structure to be executed during structure traversal. Executing a structure consists of these steps:
1. Save the current state of the rendering pipeline.
2. Set the global transform to the current composite modeling transform.
3. Set the local transform to the identity matrix.
4. Process all structure elements in the called structure.
5. Restore the state saved at step 1.

If structure does not exist at the time PEXExecuteStructure is processed, a BadPEXOutputCommand error is produced. The structure must first be created.

ERRORS
BadPEXOutputCommand
The output command contains an invalid value.
BadPExRenderer
The specified renderer resource identifier is invalid.
BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO
PEXCreateStructure(3)  
PEXBeginStructure(3)  
PEXEndStructure(3)
NAME
PEXExtendedCellArray - Extended 3D Cell Array Primitive

SYNTAX
void PEXExtendedCellArray(Display ∗display, XID resource_id, PEXOCRequestType req_type, PEXCoord ∗point1, PEXCoord ∗point2, PEXCoord ∗point3, unsigned int col_count, unsigned int row_count, int color_type, PEXArrayOfColor colors)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
point1 The first cell array definition point.
point2 The second cell array definition point.
point3 The third cell array definition point.
col_count The number of cell columns (number of cells in the X direction).
row_count The number of cell rows (number of cells in the Y direction).
color_type The type of color for the cell colors (PEXColorTypeIndexed, PEXColorTypeRGB, PEXColorTypeCIE, PEXColorTypeHSV, PEXColorTypeHLS, PEXColorTypeRGB8, PEXColorTypeRGB16).

RETURNS
None

DESCRIPTION
This function creates a 3D cell array output primitive.
This function is similar to PEXCellArray(3), except the colors are passed as either indexed color values or direct color values, depending on the color type.

DATA STRUCTURES
typedef union {
    PEXColorIndexed ∗indexed;
    PEXColorRGB ∗rgb;
    PEXColorHSV ∗hsv;
    PEXColorHLS ∗hls;
    PEXColorCIE ∗cie;
    PEXColorRGB8 ∗rgb8;
    PEXColorRGB16 ∗rgb16;
} PEXArrayOfColor;
See also PEXlib.h.

ERRORS
BadPEXOutputCommand
The output command contains an invalid value.
BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.
NAME
PEXFetchElements - Fetch Elements

SYNTAX
Status PEXFetchElements(Display *display, PEXStructure structure, int whence1, long offset1, int whence2, long offset2, int float_format, unsigned long *count_return, unsigned long *length_return, char **ocs_return)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
structure The resource identifier of the structure.
whence1 A value specifying, with offset1, the first limit of the range of elements to be fetched (PEXBeginning, PEXCurrent, PEXEnd).
offset1 The offset from whence1 denoting the first limit of the range of elements to be fetched.
whence2 A value specifying, with offset2, the second limit of the range of elements to be fetched (PEXBeginning, PEXCurrent, PEXEnd).
offset2 The offset from whence2 denoting the second limit of the range of elements to be fetched.
float_format The floating point format to use when formatting the output commands to be fetched (PEXIEEE_754_32, PEXDEC_F_Floating, PEXIEEE_754_64, PEXDEC_D_Floating).
count_return Returns the number of output commands returned.
length_return Returns the length, in bytes, of the output commands fetched.
ocs_return Returns a pointer to protocol-formatted output commands (structure elements).

RETURNS
Zero if unsuccessful, non-zero otherwise.

DESCRIPTION
This function fetches a range of structure elements from the specified structure.

If either computed offset is less than zero, it is set to zero before fetching the structure elements. If either computed offset is greater than the number of elements in the structure, it is set to the offset of the last structure element in the structure. The element pointer attribute of structure is not affected by this command. No information will be returned for inquiries on element offset zero.

An null pointer is returned if the requested floating point format is not supported.

Any text or annotation text output commands returned will be formatted as encoded text or encoded annotation text.

PEXlib allocates memory for the return value. XFree should be called to deallocate the memory.
### DATA STRUCTURES
```c
typedef XID PEXStructure;
```

### ERRORS
- **BadPEXFloatingPointFormat**
  - The specified floating point format is invalid or unsupported.
- **BadPEXStructure**
  - The specified `structure` resource identifier is invalid.
- **BadValue**
  - The specified value for whence parameter is invalid.

### SEE ALSO
- `PEXCreateStructure(3)`
- `PEXDecodeOCs(3)`
- `PEXEncodeOCs(3)`
- `PEXSendOCs(3)`
NAME PEXFetchElementsAndSend - Fetch Elements and Send to Display

SYNTAX Status PEXFetchElementsAndSend(Display *src_display, PEXStructure structure, int whence1, long offset1, int whence2, long offset2, Display *dst_display, XID resource_id, PEXOCRequestType req_type)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
structure The resource identiﬁer of the structure.
whence1 A value specifying, with offset1, the ﬁrst limit of the range of elements to be fetched ( PEXBeginning, PEXCurrent, PEXEnd ).
offset1 The offset from whence1 denoting the ﬁrst limit of the range of elements to be fetched.
whence2 A value specifying, with offset2, the second limit of the range of elements to be fetched ( PEXBeginning, PEXCurrent, PEXEnd ).
offset2 The offset from whence2 denoting the second limit of the range of elements to be fetched.
dst_display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identiﬁer of the renderer or structure.
req_type The request type for the output commands ( PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle ).

RETURNS Zero if unsuccessful, non-zero otherwise.

DESCRIPTION This function is like PEXFetchElements(3) except that the list of output commands are not returned to the application but are sent directly to the speciﬁed destination display. Calling this function is similar to calling PEXFetchElements(3), and then sending the returned list of output commands by calling PEXStartOC(3), PEXCopyBytesToOC(3) and PEXFinishOCs(3).

If the destination display does not support the same ﬂoating point format as the format PEXlib is using with the source display, and if PEXlib can not convert to a format supported by the destination display, the function will return unsuccessfully.

Sending output commands to a structure whose editing mode is PEXStructureReplace, is not really useful. The behavior will be unpredictable unless a request type of PEXOCStoreSingle is used. And, if the request type is PEXOCStoreSingle, each output command will simply replace the previous one sent. Applications should ensure that the structure’s editing mode is PEXStructureInsert, when sending multiple output commands. If it is intended to replace multiple elements, the application can delete those elements first, and then insert the new ones.

modified November 1995 101
<table>
<thead>
<tr>
<th>DATA STRUCTURES</th>
<th>typedef XID PEXStructure;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERRORS</td>
<td>BadPEXStructure</td>
</tr>
<tr>
<td></td>
<td>The specified structure resource identifier is invalid.</td>
</tr>
<tr>
<td></td>
<td>BadValue</td>
</tr>
<tr>
<td></td>
<td>The specified value for whence parameter is invalid.</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>PEXFetchElements(3)</td>
</tr>
</tbody>
</table>
NAME
PEXFillArea - 3D Fill Area Primitive

SYNTAX
void PEXFillArea(Display *display, XID resource_id, PEXOCRequestType req_type, int shape_hint, int ignore_edges, unsigned int count, PEXCoord *points)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
shape_hint The shape of the fill area (PEXShapeComplex, PEXShapeNonConvex, PEXShapeConvex, PEXShapeUnknown).
ignore_edges A flag that determines if surface edges are rendered (True or False).
count The number of points.
points An array of points defining the fill area.

RETURNS
None

DESCRIPTION
This function creates a fill area output primitive.

The area is defined by the list of points joined together to form a planar surface. Fill areas are not strictly required to be planar, but shading artifacts may occur if a fill area is not planar or nearly so. The first vertex of the fill area is connected to the second, the second to the third, and so on. The last vertex is implicitly connected to the first.

During the rendering process, the fill area vertices are transformed to positions in device coordinates. The surface colors are affected by the reflectance calculations which uses the light state, interior style, and reflection model attributes. Surface colors are further affected by the depth-cueing computation and then mapped to device colors. Fill areas outside the currently-defined clipping volume are not displayed. Fill areas crossing the clipping volume are clipped, and only the portions inside the clipping volume are displayed.

A fill area with fewer than three vertices is considered degenerate. It is stored in a structure, but when rendered, the primitive is ignored and has no visual effect.

A fill area may cross over itself to create a complex shape. The odd-even rule is used for determining the area that lies in the interior of the fill area. The shape hint is provided to enable performance improvements for certain shapes. Fill areas that are of higher complexity than indicated by the shape hint are rendered in an implementation-dependent manner. Consequently, applications should pass PEXShapeUnknown as the shape unless they are certain the fill area’s shape is one of the other three.

The ignore edges flag is a boolean value specifying whether surface edges are rendered. If the flag is True, no surface edges are rendered for the fill area. If the ignore edges flag is False, surface edges are rendered according to the surface edge attributes if the surface edge flag attribute is PEXOn. Depending on the setting of the surface edge ASF values,
the surface edges, surface edge color, surface edge type, and surface edge width attributes are obtained from one of two sources. These attributes are obtained directly from the current surface edge attribute values or from the edge bundle lookup table entry specified by the current edge bundle index attribute depending on the setting of the surface edge ASF attribute.

Depending on the setting of the surface attribute ASF values, the surface color, interior style, interior style index, surface interpolation method, and reflection model attributes are obtained from one of two sources. These attributes are obtained directly from the current surface attributes values or from the interior bundle lookup table entry specified by the current interior bundle index attribute.

When a surface is rendered, the surface color and reflection attributes are used to compute the colors of the surface if it is front-facing with respect to the point of view and the current culling mode allows front-faces to be rendered. If the surface is back-facing, the current distinguish mode is True, and the current culling mode allows back-faces to be rendered, the corresponding back-facing attributes are used instead.

Regardless of the fill area orientation, if the interior style is PEXInteriorStylePattern, the pattern size, pattern reference point and pattern reference vectors are used to pattern the fill area.

DATA STRUCTURES

See PEXlib.h.

ERRORS

BadPEXOutputCommand
The output command contains an invalid value.

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO

PEXSetInteriorStyle(3)
PEXSetInteriorStyleIndex(3)
PEXSetSurfaceColorIndex(3)
PEXSetSurfaceColor(3)
PEXSetReflectionAttributes(3)
PEXSetReflectionModel(3)
PEXSetSurfaceInterpMethod(3)
PEXSetBFInteriorStyle(3)
PEXSetBFInteriorStyleIndex(3)
PEXSetBFSurfaceColorIndex(3)
PEXSetBFSurfaceColor(3)
PEXSetBFReflectionAttributes(3)
PEXSetBFReflectionModel(3)
PEXSetBFSurfaceInterpMethod(3)
PEXSetFacetCullingMode(3)

modified November 1995
PEXSetFacetDistinguishFlag(3)
PEXSetPatternSize(3)
PEXSetPatternAttributes(3)
PEXSetPatternAttributes2D(3)
PEXSetInteriorBundleIndex(3)
PEXSetSurfaceEdgeFlag(3)
PEXSetSurfaceEdgeType(3)
PEXSetSurfaceEdgeWidth(3)
PEXSetSurfaceEdgeColor(3)
PEXSetSurfaceEdgeColorIndex(3)
PEXSetEdgeBundleIndex(3)
NAME
PEXFillArea2D - 2D Fill Area Primitive

SYNTAX
void PEXFillArea2D(Display ∗display, XID resource_id, PEXOCRequestType req_type, int shape_hint, int ignore_edges, unsigned int count, PEXCoord2D ∗points)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
shape_hint The shape of the fill area (PEXShapeComplex, PEXShapeNonConvex, PEXShapeConvex, PEXShapeUnknown).
ignore_edges A flag that determines if surface edges are rendered (True or False).
count The number of points.
points An array of points defining the fill area.

RETURNS
None

DESCRIPTION
This function creates a 2D fill area output primitive.

This function is like PEXFillArea(3), except that the vertices consist of only x- and y-components. The z-component is assumed to be zero. This primitive is two-dimensional only in that the z-components are implied. Geometry transformations are still carried out in three dimensions.

DATA STRUCTURES
See PEXlib.h.

ERRORS
BadPEXOutputCommand
The output command contains an invalid value.
BadPEXRenderer
The specified renderer resource identifier is invalid.
BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO
PEXSetInteriorStyle(3)
PEXSetInteriorStyleIndex(3)
PEXSetSurfaceColorIndex(3)
PEXSetSurfaceColor(3)
PEXSetReflectionAttributes(3)
PEXSetReflectionModel(3)
PEXSetSurfaceInterpMethod(3)
PEXSetBFIInteriorStyle(3)
PEXSetBFIInteriorStyleIndex(3)
**NAME**
PEXFillAreaSet - 3D Set of Fill Areas Primitive

**SYNTAX**

```c
void PEXFillAreaSet(Display *display, XID resource_id, PEXOCRequestType req_type, int shape_hint, int ignore_edges, int contour_hint, unsigned int count, PEXListOfCoord *point_lists)
```

**PARAMETERS**

- `display` A pointer to a display structure returned by a successful `XOpenDisplay` call.
- `resource_id` The resource identifier of the renderer or structure.
- `req_type` The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- `shape_hint` The shape which describes all of the contours (PEXShapeComplex, PEXShapeNonConvex, PEXShapeConvex, PEXShapeUnknown).
- `ignore_edges` A flag that determines if surface edges are rendered (True or False).
- `contour_hint` A flag that indicates whether contours are disjoint or overlapping (PEXContourDisjoint, PEXContourNested, PEXContourIntersecting, PEXContourUnknown).
- `count` The number of fill areas in the set.
- `point_lists` A pointer to the list of point arrays defining each contour of the fill area set.

**RETURNS**

None

**DESCRIPTION**

This function creates a fill area set output primitive.

This function is similar to PEXFillArea(3), but allows for the creation of areas with islands or holes.

If any fill area in the set has fewer than three vertices, or if there are no contours defined, the primitive is considered degenerate. The primitive is stored in a structure, but when rendered, the primitive is ignored and has no visual effect.

A fill area set consists of an array of fill areas that define "contours" (disjoint pieces or holes) making up the primitive. Each fill area, or contour, is defined by a list of vertices joined together to form a planar surface.

The contour hint provides further information about the relationships between contours in the fill area set. If the contour hint is PEXContourDisjoint, all contours will be spatially disjoint. No overlapping or intersection occurs between any contours in the fill area set. If the contour hint is PEXContourNested, contours will either be disjoint or wholly contained within another contour. No contour will have edges that intersect or are coincident with edges of any other contour. If the contour hint is PEXContourIntersecting, separated contours may have edges that are coincident or overlap. If the contour hint is PEXContourUnknown nothing is known about the interrelationships between contours. Fill area sets with contours that have higher complexity interrelationships than that indicated by the contour hint are rendered in an implementation-dependent manner.

modified November 1995
The ignore edges flag is applied to each of the fill areas in the set. All other aspects of this primitive are the same as **PEXFillArea(3)**.

**DATA STRUCTURES**

See PEXlib.h.

**ERRORS**

- **BadPEXOutputCommand**
  The output command contains an invalid value.

- **BadPEXRenderer**
  The specified renderer resource identifier is invalid.

- **BadPEXStructure**
  The specified structure resource identifier is invalid.

**SEE ALSO**

- PEXSetInteriorStyle(3)
- PEXSetInteriorStyleIndex(3)
- PEXSetSurfaceColorIndex(3)
- PEXSetSurfaceColor(3)
- PEXSetReflectionAttributes(3)
- PEXSetReflectionModel(3)
- PEXSetSurfaceInterpMethod(3)
- PEXSetBFInteriorStyle(3)
- PEXSetBFInteriorStyleIndex(3)
- PEXSetBFSurfaceColorIndex(3)
- PEXSetBFSurfaceColor(3)
- PEXSetBFReflectionAttributes(3)
- PEXSetBFReflectionModel(3)
- PEXSetBFSurfaceInterpMethod(3)
- PEXSetFacetCullingMode(3)
- PEXSetFacetDistinguishFlag(3)
- PEXSetPatternSize(3)
- PEXSetPatternAttributes(3)
- PEXSetPatternAttributes2D(3)
- PEXSetInteriorBundleIndex(3)
- PEXSetSurfaceEdgeFlag(3)
- PEXSetSurfaceEdgeType(3)
- PEXSetSurfaceEdgeWidth(3)
- PEXSetSurfaceEdgeColor(3)
- PEXSetSurfaceEdgeColorIndex(3)
- PEXSetEdgeBundleIndex(3)
NAME
PEXFillAreaSet2D - 2D Set of Fill Areas Primitive

SYNTAX
void PEXFillAreaSet2D(Display *display, XID resource_id, PEXOCRequestType req_type,
int shape_hint, int ignore_edges, int contour_hint, unsigned int count,
PEXListOfCoord2D *point_lists)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
shape_hint The shape which describes all of the contours (PEXShapeComplex, PEXShapeNonConvex, PEXShapeConvex, PEXShapeUnknown).
ignore_edges A flag that determines if surface edges are rendered (True or False).
contour_hint A flag that indicates whether contours are disjoint or overlapping (PEXContourDisjoint, PEXContourNested, PEXContourIntersecting, PEXContourUnknown).
count The number of fill areas in the set.
point_lists A pointer to the list of point arrays defining each contour of the fill area set.

RETURNS
None

DESCRIPTION
This function creates a 2D fill area set output primitive. This function is like PEXFillAreaSet(3), except that the vertices consist of only x- and y-components. The z-component is assumed to be zero. This primitive is two-dimensional only in that the z-components are implied. Geometry transformations are still carried out in three dimensions.

DATA STRUCTURES
See PEXlib.h.

ERRORS
BadPEXOutputCommand
The output command contains an invalid value.
BadPEXRenderer
The specified renderer resource identifier is invalid.
BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO
PEXSetInteriorStyle(3)
PEXSetInteriorStyleIndex(3)
PEXSetSurfaceColorIndex(3)

modified November 1995
PEXSetSurfaceColor(3)
PEXSetReflectionAttributes(3)
PEXSetReflectionModel(3)
PEXSetSurfaceInterpMethod(3)
PEXSetBFInteriorStyle(3)
PEXSetBFInteriorStyleIndex(3)
PEXSetBFSurfaceColorIndex(3)
PEXSetBFSurfaceColor(3)
PEXSetBFRFlectionAttributes(3)
PEXSetBFReflectionModel(3)
PEXSetBFSurfaceInterpMethod(3)
PEXSetFacetCullingMode(3)
PEXSetFacetDistinguishFlag(3)
PEXSetPatternSize(3)
PEXSetPatternAttributes(3)
PEXSetPatternAttributes2D(3)
PEXSetInteriorBundleIndex(3)
PEXSetSurfaceEdgeFlag(3)
PEXSetSurfaceEdgeType(3)
PEXSetSurfaceEdgeWidth(3)
PEXSetSurfaceEdgeColor(3)
PEXSetSurfaceEdgeColorIndex(3)
PEXSetEdgeBundleIndex(3)
NAME
PEXFillAreaSetWithData - 3D Set of Fill Areas Primitive With Additional Data

SYNTAX
void PEXFillAreaSetWithData(Display ∗display, XID resource_id, PEXOCRequestType
req_type, int shape_hint, int ignore_edges, int contour_hint, unsigned int
facet_attributes, unsigned int vertex_attributes, int color_type, unsigned int count,
PEXFacetData ∗facet_data, PEXListOfVertex ∗vertex_lists)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay
call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOC-
Store, PEXOCRenderSingle or PEXOCStoreSingle).
shape_hint The shape which describes all of the contours (PEXShapeComplex,
PEXShapeNonConvex, PEXShapeConvex, PEXShapeUnknown).
ignore_edges A flag that determines if surface edges are rendered (True or False).
contour_hint A flag that indicates whether contours are disjoint or overlapping
(PEXContourDisjoint, PEXContourNested, PEXContourIntersecting,
PEXContourUnknown).
facet_attributes A mask indicating the facet attributes provided (PEXGANone, PEX-
GAColor, PEXGANormal).
vertex_attributes A mask indicating the vertex attributes provided (PEXGANone, PEX-
GAColor, PEXGANormal, PEXGAEges).
color_type The type of color data provided (PEXColorTypeIndexed, PEXColor-
TypeRGB, PEXColorTypeCIE, PEXColorTypeHSV, PEXColor-
TypeHLS, PEXColorTypeRGB8, PEXColorTypeRGB16).
count The number of fill areas in the set.
facet_data A pointer to facet data.
vertex_lists A pointer to the list of vertex arrays defining each contour of the fill area
set.

RETURNS None

DESCRIPTION
This function creates a fill area set output primitive.
This function is like PEXFillAreaSet(3) except that it allows additional information to be
specified for each fill area and for each vertex. It is similar to PEXFillAreaWithData(3)
but allows for the creation of areas with islands or holes. Color values passed must be of
the specified color type.
The facet attributes indicate the content of the facet data. This data may be a color, a normal,
or a color followed by a normal. Use the constants PEXGANone, PEXGAColor and
PEXGANormal to construct a mask indicating the data provided. If specified, the facet
color takes precedence over the surface color. If specified, the facet normal is used to

modified November 1995
determine whether the fill area is back-facing.

The vertex attributes indicate the content of each fill area vertex. In addition to the coordinate \( (x,y,z) \), applications may specify a color, a normal, an edge flag, or any combination of the three specified in the order given. Use the constants \texttt{PEXGANone}, \texttt{PEXGAColor}, \texttt{PEXGANormal} and \texttt{PEXGAEdges} to construct a mask indicating the data provided. If specified, vertex colors will override facet color or the current surface color. If specified, vertex normals are taken to be normals at the vertices of the fill area.

The reflection model and the surface interpolation will affect how the additional data is used in rendering the surface. Edge controls are used to indicate those edges rendered if the surface edges are enabled. The edge control for vertex \( i \) indicates whether or not to render the edge between vertex \( i \) and vertex \( i+1 \). Surface edges are always rendered with the surface edge color and are not affected by the facet or vertex colors.

Normals are assumed to be unit length vectors. The effect if the normal is not unit length is implementation-dependent.

All other aspects of this primitive are the same as \texttt{PEXFillAreaSet(3)}.

\begin{center}
\begin{tabular}{|l|}
\hline
\textbf{DATA STRUCTURES} \\
\hline
See \texttt{PEXlib.h}. \\
\hline
\textbf{ERRORS} \\
\hline
\textbf{BadPEXOutputCommand} \\
The output command contains an invalid value. \\
\textbf{BadPEXRenderer} \\
The specified renderer resource identifier is invalid. \\
\textbf{BadPEXStructure} \\
The specified structure resource identifier is invalid. \\
\hline
\textbf{SEE ALSO} \\
\texttt{PEXSetInteriorStyle(3)} \\
\texttt{PEXSetInteriorStyleIndex(3)} \\
\texttt{PEXSetSurfaceColorIndex(3)} \\
\texttt{PEXSetSurfaceColor(3)} \\
\texttt{PEXSetReflectionAttributes(3)} \\
\texttt{PEXSetReflectionModel(3)} \\
\texttt{PEXSetSurfaceInterpMethod(3)} \\
\texttt{PEXSetBFInteriorStyle(3)} \\
\texttt{PEXSetBFInteriorStyleIndex(3)} \\
\texttt{PEXSetBFSurfaceColorIndex(3)} \\
\texttt{PEXSetBFSurfaceColor(3)} \\
\texttt{PEXSetBFReflectionAttributes(3)} \\
\texttt{PEXSetBFReflectionModel(3)} \\
\texttt{PEXSetBFSurfaceInterpMethod(3)} \\
\texttt{PEXSetFacetCullingMode(3)} \\
\texttt{PEXSetFacetDistinguishFlag(3)} \\
\texttt{PEXSetPatternSize(3)} \\
\texttt{PEXSetPatternAttributes(3)} \\
\hline
\end{tabular}
\end{center}
PEXFillAreaSetWithData(3)
PEXSetInteriorBundleIndex(3)
PEXSetSurfaceEdgeFlag(3)
PEXSetSurfaceEdgeType(3)
PEXSetSurfaceEdgeWidth(3)
PEXSetSurfaceEdgeColor(3)
PEXSetSurfaceEdgeColorIndex(3)
PEXSetEdgeBundleIndex(3)
NAME  PEXFillAreaWithData - 3D Fill Area Primitive With Additional Data

SYNTAX  void PEXFillAreaWithData(Display *display, XID resource_id, PEXOCRequestType req_type, int shape_hint, int ignore_edges, unsigned int facet_attributes, unsigned int vertex_attributes, int color_type, PEXFacetData *facet_data, unsigned int count, PEXArrayOfVertex vertices)

PARAMETERS  
- display: A pointer to a display structure returned by a successful XOpenDisplay call.
- resource_id: The resource identifier of the renderer or structure.
- req_type: The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- shape_hint: The shape of the fill area (PEXShapeComplex, PEXShapeNonConvex, PEXShapeConvex, PEXShapeUnknown).
- ignore_edges: A flag that determines if surface edges are rendered (True or False).
- facet_attributes: A mask indicating the facet attributes provided (PEXGANone, PEXGAColor, PEXGANormal).
- vertex_attributes: A mask indicating the vertex attributes provided (PEXGANone, PEXGAColor, PEXGANormal).
- color_type: The type of color data provided (PEXColorTypeIndexed, PEXColorTypeRGB, PEXColorTypeCIE, PEXColorTypeHSV, PEXColorTypeHLS, PEXColorTypeRGB8, PEXColorTypeRGB16).
- facet_data: A pointer to facet data.
- count: The number of vertices.
- vertices: An array of vertices defining the fill area.

RETURNS  None

DESCRIPTION  This function creates a fill area output primitive.

This function is like PEXFillArea(3) except that it allows additional information to be specified for the fill area and for each vertex. Color values passed must be of the specified color type.

The facet attributes indicate the content of the facet data. This data may be a color, a normal, or a color followed by a normal. Use the constants PEXGANone, PEXGAColor and PEXGANormal to construct a mask indicating the data provided. If specified, the facet color takes precedence over the surface color. If specified, the facet normal is used to determine whether the fill area is back-facing.

The vertex attributes indicate the content of each fill area vertex. In addition to the coordinate (x,y,z), applications may specify a color, a normal, or a color followed by a normal for each vertex. Use the constants PEXGANone, PEXGAColor and PEXGANormal to construct a mask indicating the data provided. If specified, vertex colors will override
facet color or the current surface color. If specified, vertex normals are taken to be normals at the vertices of the fill area.

The reflection model and the surface interpolation will affect how the additional data is used in rendering the surface.

Normals are assumed to be unit length vectors. The effect if the normal is not unit length is implementation-dependent.

All other aspects of this primitive are the same as `PEXFillArea(3)`.

**DATA STRUCTURES**

See `PEXlib.h`.

**ERRORS**

- **BadPEXOutputCommand**
  The output command contains an invalid value.

- **BadPEXRenderer**
  The specified renderer resource identifier is invalid.

- **BadPEXStructure**
  The specified structure resource identifier is invalid.

**SEE ALSO**

- `PEXSetInteriorStyle(3)`
- `PEXSetInteriorStyleIndex(3)`
- `PEXSetSurfaceColorIndex(3)`
- `PEXSetSurfaceColor(3)`
- `PEXSetReflectionAttributes(3)`
- `PEXSetReflectionModel(3)`
- `PEXSetSurfaceInterpMethod(3)`
- `PEXSetBFInteriorStyle(3)`
- `PEXSetBFInteriorStyleIndex(3)`
- `PEXSetBFSurfaceColorIndex(3)`
- `PEXSetBFSurfaceColor(3)`
- `PEXSetBFReflectionAttributes(3)`
- `PEXSetBFReflectionModel(3)`
- `PEXSetBFSurfaceInterpMethod(3)`
- `PEXSetFacetCullingMode(3)`
- `PEXSetFacetDistinguishFlag(3)`
- `PEXSetPatternSize(3)`
- `PEXSetPatternAttributes(3)`
- `PEXSetPatternAttributes2D(3)`
- `PEXSetInteriorBundleIndex(3)`
- `PEXSetSurfaceEdgeFlag(3)`
- `PEXSetSurfaceEdgeType(3)`
- `PEXSetSurfaceEdgeWidth(3)`
- `PEXSetSurfaceEdgeColor(3)`
- `PEXSetSurfaceEdgeColorIndex(3)`
- `PEXSetEdgeBundleIndex(3)`

Modified November 1995
### NAME
PEXFinishOCs – Finish Encoded Output Commands

### SYNTAX
```c
void PEXFinishOCs(Display *display)
```

### PARAMETERS
- **display**: A pointer to a display structure returned by a successful `XOpenDisplay` call.

### RETURNS
None

### DESCRIPTION
This function should be called to complete the sending of encoded output commands. The display connection is unlocked.

### ERRORS
None

### SEE ALSO
- PEXStartOCs(3)
- PEXCopyBytesToOC(3)
- PEXGetOCAddr(3)
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXFreeEnumInfo – Free Memory Allocated for Enumerated Type Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td>void PEXFreeEnumInfo(unsigned long count, unsigned long *info_count, PEXEnumTypeDesc *enum_info)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>count: The number of enumerated types.</td>
</tr>
<tr>
<td></td>
<td>info_count: An array of counts. This corresponds to the info_count_return parameter in PEXGetEnumTypeInfo(3).</td>
</tr>
<tr>
<td></td>
<td>enum_info: An array of enumerated type descriptors. This corresponds to the enum_info_return parameter of PEXGetEnumTypeInfo(3).</td>
</tr>
<tr>
<td>RETURNS</td>
<td>None</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>PEXFreeEnumInfo frees memory allocated by PEXlib for the return value and return parameter in PEXGetEnumTypeInfo(3).</td>
</tr>
<tr>
<td>ERRORS</td>
<td>None</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>PEXGetEnumTypeInfo(3)</td>
</tr>
</tbody>
</table>
NAME
PEXFreeFontInfo – Free Font Info Returned by PEXListFontsWithInfo and PEXQueryFont

SYNTAX
void PEXFreeFontInfo(unsigned long \textit{count}, PEXFontInfo *\textit{font_info})

PARAMETERS
\textit{count} \hspace{1cm} The number of font info structures.
\textit{font_info} \hspace{1cm} An array of font info structures.

RETURNS
None

DESCRIPTION
This function deallocates memory returned by \texttt{PEXListFontsWithInfo(3)} and \texttt{PEXQueryFont(3)}.

ERRORS
None

SEE ALSO
\texttt{PEXListFontsWithInfo(3)}
\texttt{PEXQueryFont(3)}

modified November 1995
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXFreeFontNames – Free Font Names Returned by PEXListFonts, PEXListFontsWithInfo</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td>void PEXFreeFontNames(unsigned long count, char **font_names)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>count: The number of font names.</td>
</tr>
<tr>
<td></td>
<td>font_names: An array of font names (null-terminated strings).</td>
</tr>
<tr>
<td>RETURNS</td>
<td>None</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>This function deallocates memory returned by PEXListFonts(3) and PEXListFontsWithInfo(3).</td>
</tr>
<tr>
<td>ERRORS</td>
<td>None</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>PEXListFonts(3)</td>
</tr>
<tr>
<td></td>
<td>PEXListFontsWithInfo(3)</td>
</tr>
</tbody>
</table>
NAME        PEXFreeLookupTable – Free Lookup Table
SYNTAX      void PEXFreeLookupTable(Display *display, PEXLookupTable table)
PARAMETERS  
display      A pointer to a display structure returned by a successful XOpenDisplay call.

table       The resource identifier of the lookup table.
RETURNS     None
DESCRIPTION  This function deletes the association between the lookup table resource identifier and the lookup table. The lookup table is freed when no other resource references it.

ERRORS      BadPEXLookupTable
            The specified lookup table resource identifier is invalid.
SEE ALSO    PEXCreateLookupTable(3)
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXFreeNameSet – Free Name Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td>void PEXFreeNameSet(Display *display, PEXNameSet nameset)</td>
</tr>
</tbody>
</table>
| PARAMETERS |  display | A pointer to a display structure returned by a successful XOpenDisplay call.  
<p>| | nameset | The resource identifier of the name set. |
| RETURNS | None |
| DESCRIPTION | This function deletes the association between the name set resource identifier and the name set. The name set is freed when no other resource references it. |
| ERRORS | BadPEXNameSet | The specified name set resource identifier is invalid. |
| SEE ALSO | PEXCreateNameSet(3) |</p>
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXFreeOCData – Deallocate OC Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td>void PEXFreeOCData(unsigned long count, PEXOCData *oc_data)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>count The number of output commands.</td>
</tr>
<tr>
<td></td>
<td>oc_data An array of output command data.</td>
</tr>
<tr>
<td>RETURNS</td>
<td>None</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>This function deallocates memory allocated by PEXlib to hold decoded output command data.</td>
</tr>
<tr>
<td>ERRORS</td>
<td>None</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>PEXDecodeOCs(3)</td>
</tr>
<tr>
<td>NAME</td>
<td>PEXFreePCAttributes – Free Storage Returned by PEXGetPipelineContext</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------------------------</td>
</tr>
<tr>
<td>SYNTAX</td>
<td>void PEXFreePCAttributes(PEXPCAttributes *values)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>values A pointer to the pipeline context attribute values.</td>
</tr>
<tr>
<td>RETURNS</td>
<td>None</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>This function deallocates memory returned by PEXGetPipelineContext.</td>
</tr>
<tr>
<td>ERRORS</td>
<td>None</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>PEXGetPipelineContext(3)</td>
</tr>
<tr>
<td>NAME</td>
<td>PEXFreePDAttributes – Free Storage Returned by PEXGetPickDevice</td>
</tr>
<tr>
<td>------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>SYNTAX</td>
<td><code>void PEXFreePDAttributes(PEXPDAtributes *values)</code></td>
</tr>
<tr>
<td>PARAMETERS</td>
<td><code>values</code> A pointer to the pick device attribute values.</td>
</tr>
<tr>
<td>RETURNS</td>
<td>None</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>This function deallocates memory returned by <code>PEXGetPickDevice(3)</code>.</td>
</tr>
<tr>
<td>ERRORS</td>
<td>None</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td><code>PEXGetPickDevice(3)</code></td>
</tr>
<tr>
<td><strong>NAME</strong></td>
<td>PEXFreePMAttributes – Free Storage Returned by PEXGetPickMeasure</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>SYNTAX</strong></td>
<td>void PEXFreePMAttributes(PEXPMAttributes *values)</td>
</tr>
<tr>
<td><strong>PARAMETERS</strong></td>
<td>values A pointer to the pick measure attribute values.</td>
</tr>
<tr>
<td><strong>RETURNS</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>DESCRIPTION</strong></td>
<td>This function deallocates memory returned by PEXGetPickMeasure(3).</td>
</tr>
<tr>
<td><strong>ERRORS</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>SEE ALSO</strong></td>
<td>PEXGetPickMeasure(3)</td>
</tr>
<tr>
<td><strong>NAME</strong></td>
<td>PEXFreePickMeasure – Free Pick Measure</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td><strong>SYNTAX</strong></td>
<td>void PEXFreePickMeasure(Display *display, PEXPickMeasure pick_measure)</td>
</tr>
<tr>
<td><strong>PARAMETERS</strong></td>
<td>display A pointer to a display structure returned by a successful XOpenDisplay call.</td>
</tr>
<tr>
<td></td>
<td>pick_measure The resource identifier of the pick measure.</td>
</tr>
<tr>
<td><strong>RETURNS</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>DESCRIPTION</strong></td>
<td>This function deletes the pick measure resource and frees memory associated with it.</td>
</tr>
<tr>
<td><strong>ERRORS</strong></td>
<td>BadPEXPickMeasure The specified pick measure resource identifier is invalid.</td>
</tr>
<tr>
<td><strong>SEE ALSO</strong></td>
<td>PEXCreatePickMeasure(3)</td>
</tr>
</tbody>
</table>
NAME       PEXFreePickPaths – Free Memory Allocated for Pick Paths
SYNTAX    void PEXFreePickPaths(unsigned long count, PEXPickPath *pick_paths)
PARAMETERS count The number of pick paths.
pick_paths An array of pick paths.
RETURNS None
DESCRIPTION This function deallocates memory returned by PEXEndPickAll(3) and PEXPickAll(3).
ERRORS None
SEE ALSO PEXEndPickAll(3)
PEXPickAll(3)
NAME
PEXFreePipelineContext – Free Pipeline Context

SYNTAX
void PEXFreePipelineContext(Display *display, PEXPipelineContext context)

PARAMETERS
display    A pointer to a display structure returned by a successful XOpenDisplay call.
context    The resource identifier of the pipeline context.

RETURNS
None

DESCRIPTION
This function deletes the association between the pipeline context resource identifier and the pipeline context. The pipeline context is freed when no other resource references it.

ERRORS
BadPEXPipelineContext
The specified pipeline context resource identifier is invalid.

SEE ALSO
PEXCreatePipelineContext(3)
### NAME
PEXFreeRenderer – Free Renderer

### SYNTAX
```c
void PEXFreeRenderer(Display *display, PEXRenderer renderer)
```

### PARAMETERS
- `display` A pointer to a display structure returned by a successful `XOpenDisplay` call.
- `renderer` The resource identifier of the renderer.

### RETURNS
None

### DESCRIPTION
This function deletes the renderer resource specified by `renderer` and frees the memory associated with it.

### ERRORS
- `BadPEXRenderer` The specified renderer resource identifier is invalid.

### SEE ALSO
- `PEXCreateRenderer(3)`

modified November 1995
### NAME
PEXFreeRendererAttributes - Free Storage Returned by PEXGetRendererAttributes

### SYNTAX
```c
void PEXFreeRendererAttributes(PEXRendererAttributes *values)
```

### PARAMETERS
- `values` - A pointer to the renderer attribute values.

### RETURNS
None

### DESCRIPTION
This function deallocates memory returned by `PEXGetRendererAttributes(3)`.

### ERRORS
None

### SEE ALSO
- `PEXGetRendererAttributes(3)`
<table>
<thead>
<tr>
<th><strong>NAME</strong></th>
<th>PEXFreeSCAttributes – Free Storage Returned by PEXGetSearchContext</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SYNTAX</strong></td>
<td>void PEXFreeSCAttributes(PEXSCAttributes *values)</td>
</tr>
<tr>
<td><strong>PARAMETERS</strong></td>
<td>values A pointer to the search context attribute values.</td>
</tr>
<tr>
<td><strong>RETURNS</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>DESCRIPTION</strong></td>
<td>This function deallocates memory returned by PEXGetSearchContext(3).</td>
</tr>
<tr>
<td><strong>ERRORS</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>SEE ALSO</strong></td>
<td>PEXGetSearchContext(3)</td>
</tr>
<tr>
<td>NAME</td>
<td>PEXFreeSearchContext – Free Search Context</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>SYNTAX</td>
<td><code>void PEXFreeSearchContext(Display *display, PEXSearchContext context)</code></td>
</tr>
</tbody>
</table>
| PARAMETERS    | **display** A pointer to a display structure returned by a successful `XOpenDisplay` call.  
**context** The resource identifier of the search context. |
| RETURNS       | None |
| DESCRIPTION   | This function deletes the search context resource and frees memory associated with it. |
| ERRORS        | **BadPEXSearchContext**  
The specified search context resource identifier is invalid. |
<p>| SEE ALSO      | PEXCreateSearchContext(3) |</p>
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXFreeStructurePaths – Free Structure Paths Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td>void PEXFreeStructurePaths(unsigned long count, PEXStructurePath *paths)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>count The number of structure paths.</td>
</tr>
<tr>
<td></td>
<td>paths An array of structure paths.</td>
</tr>
<tr>
<td>RETURNS</td>
<td>None</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>This function deallocates memory returned by PEXGetAncestors(3), PEXGetDescendants(3), and PEXSearchNetwork(3).</td>
</tr>
<tr>
<td>ERRORS</td>
<td>None</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>PEXGetAncestors(3)</td>
</tr>
<tr>
<td></td>
<td>PEXGetDescendants(3)</td>
</tr>
<tr>
<td></td>
<td>PEXSearchNetwork(3)</td>
</tr>
<tr>
<td>NAME</td>
<td>PEXFreeTableEntries – Free Storage associated with Tables Entries Return Parameters</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SYNTAX</td>
<td>void PEXFreeTableEntries(int *table_type, unsigned int count, PEXPointer entries)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td></td>
</tr>
<tr>
<td>table_type</td>
<td>The type of table entries in the array.</td>
</tr>
<tr>
<td>count</td>
<td>The number of entries in the array.</td>
</tr>
<tr>
<td>entries</td>
<td>An array of table entries.</td>
</tr>
<tr>
<td>RETURNS</td>
<td>None</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>This function deallocates memory returned by PEXGetPredefinedEntries(3), PEXGetTableEntries(3) and PEXGetTableEntry(3).</td>
</tr>
<tr>
<td>ERRORS</td>
<td>None</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>PEXGetPredefinedEntries(3)</td>
</tr>
<tr>
<td></td>
<td>PEXGetTableEntries(3)</td>
</tr>
<tr>
<td></td>
<td>PEXGetTableEntry(3)</td>
</tr>
</tbody>
</table>

modified November 1995
### NAME
PEXFreeWorkstation – Free Workstation

### SYNTAX
```c
void PEXFreeWorkstation(Display *display, PEXWorkstation workstation)
```

### PARAMETERS
- **display**: A pointer to a display structure returned by a successful `XOpenDisplay` call.
- **workstation**: The resource identifier of the workstation.

### RETURNS
None

### DESCRIPTION
This function deletes the PHIGS workstation resource and frees memory associated with it.

### ERRORS
- **BadPEXWorkstation**
  The specified `workstation` resource identifier is invalid.

### SEE ALSO
PEXCreateWorkstation(3)

---

Modified November 1995
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXFreeWorkstationAttributes – Free Storage Returned by PEXGetWorkstationAttributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td>void PEXFreeWorkstationAttributes(PEXWorkstationAttributes *values)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>values</td>
</tr>
<tr>
<td>RETURNS</td>
<td>None</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>This function deallocates memory returned by PEXGetWorkstationAttributes(3).</td>
</tr>
<tr>
<td>ERRORS</td>
<td>None</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>PEXGetWorkstationAttributes(3)</td>
</tr>
</tbody>
</table>
NAME
PEXGDP - 3D Generalized Drawing Primitive

SYNTAX
void PEXGDP(Display *display, XID resource_id, PEXOCRequestType req_type, long gdp_id, unsigned int count, PEXCoord *points, unsigned long length, char *data)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
gdp_id The identifier of the GDP.
count The number of points.
points The points used by the GDP.
length The length, in bytes, of the data.
data Additional data used by the GDP.

RETURNS None

DESCRIPTION
This function creates a generalized drawing primitive. The complete interface and behavior for each GDP identifier should be available with the individual PEX server implementations. There are no standard PEX GDPs. If the specified GDP identifier is not supported, then the output command is ignored. The table below lists the supported GDP identifier.

<table>
<thead>
<tr>
<th>GDP Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEXSunGDP3DIdCircle</td>
<td>Circle</td>
</tr>
<tr>
<td>PEXSunGDP3DIdCircArc</td>
<td>Circular arc</td>
</tr>
<tr>
<td>PEXSunGDP3DIdCircArcClose</td>
<td>Circular arc close</td>
</tr>
<tr>
<td>PEXSunGDP3DIdAnnotCircle</td>
<td>Annotation circle</td>
</tr>
<tr>
<td>PEXSunGDP3DIdAnnotCircArc</td>
<td>Annotation circle arc</td>
</tr>
<tr>
<td>PEXSunGDP3DIdAnnotCircArcClose</td>
<td>Annotation circle arc close</td>
</tr>
<tr>
<td>PEXSunGDP3DIdEllipse</td>
<td>Ellipse</td>
</tr>
<tr>
<td>PEXSunGDP3DIdEllpArc</td>
<td>Elliptical arc</td>
</tr>
<tr>
<td>PEXSunGDP3DIdEllpArcClose</td>
<td>Elliptical arc close</td>
</tr>
<tr>
<td>PEXSunGDP3DIdAnnotEllipse</td>
<td>Annotation ellipse</td>
</tr>
<tr>
<td>PEXSunGDP3DIdAnnotEllpArc</td>
<td>Annotation elliptical arc</td>
</tr>
<tr>
<td>PEXSunGDP3DIdAnnotEllpArcClose</td>
<td>Annotation elliptical arc close</td>
</tr>
<tr>
<td>PEXSunGDP3DIdRectGrid</td>
<td>Rectangular grid</td>
</tr>
<tr>
<td>PEXSunGDP3DIdRadialGrid</td>
<td>Radial grid</td>
</tr>
<tr>
<td>PEXSunGDP3DIdTriangleList</td>
<td>Triangle list</td>
</tr>
<tr>
<td>ESGdpSphere</td>
<td>Sphere</td>
</tr>
<tr>
<td>ESGdpSphereRadius</td>
<td>Sphere with radius</td>
</tr>
<tr>
<td>ESGdpSphereColour</td>
<td>Sphere with colour</td>
</tr>
</tbody>
</table>

modified November 1995
**DATA STRUCTURES**

See PEXlib.h.

**ERRORS**

**BadPEXOutputCommand**

The output command contains an invalid value.

**BadPEXRenderer**

The specified renderer resource identifier is invalid.

**BadPEXStructure**

The specified structure resource identifier is invalid.

---

<table>
<thead>
<tr>
<th>GDP Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESGdpSphereRadiusColour</td>
<td>Sphere with radius and colour</td>
</tr>
<tr>
<td>ESGdpCylinder</td>
<td>Cylinder</td>
</tr>
<tr>
<td>ESGdpCylinderRadius</td>
<td>Cylinder with radius</td>
</tr>
<tr>
<td>ESGdpCylinderColour</td>
<td>Cylinder with colour</td>
</tr>
<tr>
<td>ESGdpCylinderRadiusColour</td>
<td>Cylinder with radius and colour</td>
</tr>
</tbody>
</table>
NAME  PEXGDP2D - 2D Generalized Drawing Primitive

SYNTAX  void PEXGDP2D(Display *display, XID resource_id, PEXOCRequestType req_type, long gdp_id, unsigned int count, PEXCoord2D *points, unsigned long length, char *data)

PARAMETERS  display  A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id  The resource identifier of the renderer or structure.
req_type  The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
gdp_id  The identifier of the GDP.
count  The number of points.
points  The points used by the GDP.
length  The length, in bytes, of the data.
data  Additional data used by the GDP.

RETURNS  None

DESCRIPTION  This function creates a 2D generalized drawing primitive.
The complete interface and behavior for each GDP identifier should be available with the individual PEX server implementations. There are no standard PEX GDPs. If the specified GDP identifier is not supported, then the output command is ignored. The table below lists the supported GDP identifier.

<table>
<thead>
<tr>
<th>GDP Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEXSunGDP2DIdCircle</td>
<td>Circle</td>
</tr>
<tr>
<td>PEXSunGDP2DIdCircArc</td>
<td>Circular arc</td>
</tr>
<tr>
<td>PEXSunGDP2DIdCircArcClose</td>
<td>Circular arc close</td>
</tr>
<tr>
<td>PEXSunGDP2DIdAnnotCircle</td>
<td>Annotation circle</td>
</tr>
<tr>
<td>PEXSunGDP2DIdAnnotCircArc</td>
<td>Annotation circle arc</td>
</tr>
<tr>
<td>PEXSunGDP2DIdAnnotCircArcClose</td>
<td>Annotation circle arc close</td>
</tr>
<tr>
<td>PEXSunGDP2DIdEllipse</td>
<td>Ellipse</td>
</tr>
<tr>
<td>PEXSunGDP2DIdEllpArc</td>
<td>Elliptical arc</td>
</tr>
<tr>
<td>PEXSunGDP2DIdEllpArcClose</td>
<td>Elliptical arc close</td>
</tr>
<tr>
<td>PEXSunGDP2DIdAnnotEllipse</td>
<td>Annotation ellipse</td>
</tr>
<tr>
<td>PEXSunGDP2DIdAnnotEllpArc</td>
<td>Annotation elliptical arc</td>
</tr>
<tr>
<td>PEXSunGDP2DIdAnnotEllpArcClose</td>
<td>Annotation elliptical arc close</td>
</tr>
<tr>
<td>PEXSunGDP2DIdRectGrid</td>
<td>Rectangular grid</td>
</tr>
<tr>
<td>PEXSunGDP2DIdRadialGrid</td>
<td>Radial grid</td>
</tr>
</tbody>
</table>
See PEXlib.h.

**ERRORS**

- **BadPEXOutputCommand**
  The output command contains an invalid value.

- **BadPEXRenderer**
  The specified renderer resource identifier is invalid.

- **BadPEXStructure**
  The specified structure resource identifier is invalid.
NAME
PEXGSE - Generalized Structure Element

SYNTAX
void PEXGSE(Display *display, XID resource_id, PEXOCRequestType req_type, long id, int length, char *data)

PARAMETERS
- display: A pointer to a display structure returned by a successful XOpenDisplay call.
- resource_id: The resource identifier of the renderer or structure.
- req_type: The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- id: The identifier of the GSE.
- length: The length, in bytes, of the GSE data.
- data: A pointer to the GSE data.

RETURNS
None

DESCRIPTION
This function creates a generalized structure element.

The complete interface and behavior for each GSE identifier should be available with the individual PEX server implementations. If the specified GSE identifier is not supported, then the output command is ignored. The table below lists the supported GSE identifiers.

<table>
<thead>
<tr>
<th>GSE Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEXSunGSEIdSetHighlightColor</td>
<td>Set highlighting color</td>
</tr>
<tr>
<td>PEXSunGSEIdSetTextSlantAngle</td>
<td>Set text slant angle</td>
</tr>
<tr>
<td>PEXSunGSEIdSetAnnotTextSlantAngle</td>
<td>Set annotation text slant angle</td>
</tr>
<tr>
<td>PEXSunGSEIdSetStrokeAAliasParams</td>
<td>Set stroke anti-aliasing parameters</td>
</tr>
<tr>
<td>PEXSunGSEIdSetEndcap</td>
<td>Set stroke end cap</td>
</tr>
<tr>
<td>PEXSunGSEIdSetStrokeJoin</td>
<td>Set stroke join</td>
</tr>
<tr>
<td>PEXSunGSEIdSetSilhouetteEdgeFlag</td>
<td>Set silhouette edge flag</td>
</tr>
<tr>
<td>PEXSunGSEIdSetSurfTranspCoef</td>
<td>Set surface transparency coefficient</td>
</tr>
<tr>
<td>ESGseSphereRadius</td>
<td>Set sphere radius</td>
</tr>
<tr>
<td>ESGseSphereDivisions</td>
<td>Set sphere precision</td>
</tr>
<tr>
<td>ESGseCylinderRadius</td>
<td>Set cylinder radius</td>
</tr>
<tr>
<td>ESGseCylinderDivisions</td>
<td>Set cylinder precision</td>
</tr>
</tbody>
</table>
ERRORS

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.
NAME
PEXGSE-ESGseCylinderDivisions - E & S Cylinder Division GSE

SYNTAX
void PEXGSE(Display *display, XID resource_id, PEXOCRequestType req_type, long id,
int length, char *data)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
id The identifier of the GSE.
length The length, in bytes, of the GSE data.
data A pointer to the GSE data.

RETURNS None

DESCRIPTION
To use the E & S CylinderDivisions GSE, the id parameter should be set to ESGseCylinderDivisions, the data parameter should be set to point to structure type esCylinderDivisionData, and the length parameter should be set to the total length of data.

The E & S Cylinder Division GSE specifies the division with which cylinders are drawn. Cylinders will be drawn with div number of latitude lines and a corresponding number of longitude lines.

DATA STRUCTURES
The data structure for this GSE is defined in ESproto.h as follows:

typedef struct {
    CARD32 div;
} esCylinderDivisionData;

SEE ALSO
PEXGSE(3)

modified November 1995
NAME
PEXGSE-ESGseCylinderRadius - E & S Cylinder Radius GSE

SYNTAX
void PEXGSE(Display *display, XID resource_id, PEXOCRequestType req_type, long id, int length, char *data)

PARAMETERS
display  A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id  The resource identifier of the renderer or structure.
req_type  The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
id  The identifier of the GSE.
length  The length, in bytes, of the GSE data.
data  A pointer to the GSE data.

RETURNS
None

DESCRIPTION
To use the E & S CylinderRadius GSE, the id parameter should be set to ESgseCylinderRadius, the data parameter should be set to point to structure type esCylinderRadiusData, and the length parameter should be set to the total length of data.

Sphere GSEs that do not include a specification for radii will be rendered using an inherited radius attribute. This attribute is set in the workstation state or renderer state via the E & S Cylinder Radius GSE.

DATA STRUCTURES
The data structure for this GSE is defined in ESproto.h as follows:

typedef struct {
    PEXFLOAT radius;
} esCylinderRadiusData;

SEE ALSO
PEXGSE(3)

modified November 1995
NAME PEXGSE-ESGseSphereDivisions - E & S Sphere Division GSE

SYNTAX

```c
void PEXGSE(Display *display, XID resource_id, PEXOCRequestType req_type, long id,
int length, char *data)
```

PARAMETERS

display A pointer to a display structure returned by a successful `XOpenDisplay`
call.

resource_id The resource identifier of the renderer or structure.

req_type The request type for the output command (PEXOCRend, PEXOC-
Store, PEXOCRendSingle or PEXOCStoreSingle).

id The identifier of the GSE.

length The length, in bytes, of the GSE data.

data A pointer to the GSE data.

RETURNS None

DESCRIPTION To use the E & S SphereDivisions GSE, the `id` parameter should be set to
ESGseSphereDivisions, the `data` parameter should be set to point to structure type
esSphereDivisionData, and the `length` parameter should be set to the total length of `data`.

This GSE specifies the division with which spheres are drawn. Spheres will be drawn
with `div` number of latitude lines and a corresponding number of longitude lines.

DATA STRUCTURES The data structure for this GSE is defined in `ESproto.h` as follows:

```c
typedef struct {
    CARD32 div;
} esSphereDivisionData;
```

SEE ALSO PEXGSE(3)
NAME                  PEXGSE-ESGseSphereRadius - E & S Sphere Radius GSE

SYNTAX                void PEXGSE(Display *display, XID resource_id, PEXOCRequestType req_type, long id,
                    int length, char *data)

PARAMETERS            display    A pointer to a display structure returned by a successful XOpenDisplay
                        call.
resource_id           The resource identifier of the renderer or structure.
req_type               The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or
                        PEXOCStoreSingle).
id                    The identifier of the GSE.
length                The length, in bytes, of the GSE data.
data                  A pointer to the GSE data.

RETURNS               None

DESCRIPTION           To use the E & S SphereRadius GSE, the id parameter should be set to ESGseSphereRadius, the data parameter
                        should be set to point to structure type esSphereRadiusData, and the length parameter should be set to the total length of data.
                        Sphere GSEs that do not include a specification for radii will be rendered using an inherited radius attribute. This attribute is set in
                        the workstation state or renderer state via the E & S Sphere Radius GSE.

DATA STRUCTURES       The data structure for this GSE is defined in ESproto.h as follows:
typedef struct {
    PEXFLOAT     radius;
} esSphereRadiusData;

SEE ALSO              PEXGSE(3)
**NAME**  
PEXGSE-PEXSunGseIdSetAnnotTextSlantAngle - PEX Set Annotation Text Slant Angle

**SYNTAX**  
`void PEXGSE(Display ∗display, XID resource_id, PEXOCRequestType req_type, long id, int length, char ∗data)`

**PARAMETERS**  
- `display`  
  A pointer to a display structure returned by a successful `XOpenDisplay` call.
- `resource_id`  
  The resource identifier of the renderer or structure.
- `req_type`  
  The request type for the output command (`PEXOCRender`, `PEXOCStore`, `PEXOCRenderSingle` or `PEXOCStoreSingle`).
- `id`  
  The identifier of the GSE.
- `length`  
  The length, in bytes, of the GSE data.
- `data`  
  A pointer to the GSE data.

**RETURNS**  
None

**DESCRIPTION**  
To use the Sun SetAnnotationTextSlantAngle GSE, the `id` parameter should be set to `PEXSunGSEIdSetAnnotTextSlantAngle`, the `data` parameter should be set to point to structure type `pexSunGseAnnotTextSlantAngle`, and the `length` parameter should be set to the total length of `data`.

This GSE causes the vertical component of subsequent annotation text output commands to be slanted from the character up vector by `slant_angle`.

The slant angle, in radians, can take values between `-\pi/2` and `+\pi/2`. A negative slant angle will slant the characters in the backward direction.

**DATA STRUCTURES**  
```c
typedef struct {
    PEXFLOAT slant_angle;
} pexSunGseAnnotTextSlantAngle;
```

**SEE ALSO**  
PEXGSE(3)
NAME PEXGSE-PEXSunGseIdSetEndcap - PEX Set Endcap

SYNTAX
void PEXGSE(Display *display, XID resource_id, PEXOCRequestType req_type, long id, int length, char *data)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
id The identifier of the GSE.
length The length, in bytes, of the GSE data.
data A pointer to the GSE data.

RETURNS None

DESCRIPTION To use the Sun SetEndcap GSE, the id parameter should be set to PEXSunGSEIdSetEndcap, the data parameter should be set to point to structure type pexSunGseEndcap, and the length parameter should be set to the total length of data.

This GSE specifies the style of stroke end caps.

DATA STRUCTURES
typedef struct {
CARD32 prim;
INT32 type;
} pexSunGseEndcap;

The prim field specifies which primitives this endcap-style applies to. Values for prim are defined in SunPEX.h as listed below:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEXSunEndcapPrimLine</td>
<td>1</td>
</tr>
<tr>
<td>PEXSunEndcapPrimEdge</td>
<td>4</td>
</tr>
</tbody>
</table>

The type field specifies the endcap-style. Values for type are defined in SunPEX.h as listed below:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEXSunEndcapTypeButt</td>
<td>0</td>
</tr>
<tr>
<td>PEXSunEndcapTypeSquare</td>
<td>1</td>
</tr>
<tr>
<td>PEXSunEndcapTypeRound</td>
<td>2</td>
</tr>
</tbody>
</table>

SEE ALSO PEXGSE(3)
NAME PEXGSE-PEXSunGseIdSetHighlightColor - PEX Set Highlight Color

SYNTAX
void PEXGSE(Display *display, XID resource_id, PEXOCRequestType req_type, long id, int length, char *data)

PARAMETERS
- display: A pointer to a display structure returned by a successful XOpenDisplay call.
- resource_id: The resource identifier of the renderer or structure.
- req_type: The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- id: The identifier of the GSE.
- length: The length, in bytes, of the GSE data.
- data: A pointer to the GSE data.

RETURNS None

DESCRIPTION
To use the Sun SetHighlightColor GSE, the id parameter should be set to PEXSunGSEIdSetHighlightColor, the data parameter should be set to point to structure type pexSunGseHighlightColor, and the length parameter should be set to the total length of data.

This GSE sets the highlighting color to be used when highlighting subsequent output commands, overriding the color set by the other attributes.

DATA STRUCTURES
typedef struct {
pexColorSpeciﬁer colorspec;
    /* SINGLE COLOR() */
} pexSunGseHighlightColor;

SEE ALSO PEXGSE(3)
NAME
PEXGSE-PEXSunGseIdSetSilhouetteEdgeFlag - PEX Set Silhouette Edge Flag

SYNTAX
void PEXGSE(Display *display, XID resource_id, PEXOCRequestType req_type, long id, int length, char *data)

PARAMETERS
display - A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id - The resource identifier of the renderer or structure.
req_type - The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
id - The identifier of the GSE.
length - The length, in bytes, of the GSE data.
data - A pointer to the GSE data.

RETURNS
None

DESCRIPTION
To use the Sun SetSilhouetteEdgeFlag GSE, the id parameter should be set to PEXSunGSEIdSetSilhouetteEdgeFlag, the data parameter should be set to point to structure type pexSunGseSilhouettedEdge, and the length parameter should be set to the total length of data.

If the value of flag is ON, the server will attempt to render the silhouette edges, if any, implicit in area-filling output commands.

DATA STRUCTURES
typedef struct {
    INT32 flag;
} pexSunGseSilhouettedEdge;

Values defined for flag in SunPEX.h are listed below:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEXSunSilhouetteModeOff</td>
<td>0</td>
</tr>
<tr>
<td>PEXSunSilhouetteModeOn</td>
<td>1</td>
</tr>
</tbody>
</table>

SEE ALSO
PEXGSE(3)
NAME

PEXGSE-PEXSunGseIdSetStrokeAAliasParams - PEX Set Stroke Anti-Aliasing Parameters

SYNTAX

void PEXGSE(Display *display, XID resource_id, PEXOCRequestType req_type, long id, int length, char *data)

PARAMETERS

display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRendersonSingle or PEXOCStoreSingle).
id The identifier of the GSE.
length The length, in bytes, of the GSE data.
data A pointer to the GSE data.

RETURNS

None

DESCRIPTION

To use the Sun SetStrokeAntiAliasingParameters GSE, the id parameter should be set to PEXSunGSEIdSetStrokeAAliasParams, the data parameter should be set to point to structure type pexSunGseStrokeAAlias, and the length parameter should be set to the total length of data.

This GSE is used to control stroke anti-aliasing for subsequent stroke primitives of width equal to one.

DATA STRUCTURES

typedef struct {
    CARD32 prim;
    INT32 blend_eq;
    INT32 filter_width;
    INT32 filter_shape;
} pexSunGseStrokeAAlias;

The prim field specifies the type of primitive for which this anti-aliasing specification applies. Values for prim are defined in SunPEX.h as listed below:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEXSunAAliasPrimLine</td>
<td>0x00000001</td>
<td>Anti-alias all polyline primitives.</td>
</tr>
<tr>
<td>PEXSunAAliasPrimMarker</td>
<td>0x00000002</td>
<td>Anti-alias all polymarker primitives.</td>
</tr>
<tr>
<td>PEXSunAAliasPrimText</td>
<td>0x00000004</td>
<td>Anti-alias all text primitives.</td>
</tr>
<tr>
<td>PEXSunAAliasPrimHollowSurf</td>
<td>0x00000008</td>
<td>Anti-alias hollow surfaces.</td>
</tr>
<tr>
<td>PEXSunAAliasPrimEdge</td>
<td>0x00000010</td>
<td>Anti-alias all edges.</td>
</tr>
<tr>
<td>PEXSunAAliasPrimAll</td>
<td>0xffffffff</td>
<td>Anti-alias all of the above.</td>
</tr>
</tbody>
</table>

modified November 1995
Values for `blend_eq` are defined in `SunPEX.h` as follows:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEXSunAAliasBlendNone</td>
<td>0</td>
<td>Do not anti-alias.</td>
</tr>
<tr>
<td>PEXSunAAliasBlendArbitBG</td>
<td>1</td>
<td>Blend to an arbitrary background, blending on each pixel.</td>
</tr>
<tr>
<td>PEXSunAAliasBlendConstBG</td>
<td>2</td>
<td>Blend to a constant background color 0.</td>
</tr>
<tr>
<td>PEXSunAAliasBlendAddToBG</td>
<td>3</td>
<td>Blend by adding to the background.</td>
</tr>
</tbody>
</table>

The following value is defined for `filter_shape` in `SunPEX.h` as follows:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEXSunAAliasFiltShapeGaussian</td>
<td>0</td>
<td>Use a Gaussian filter.</td>
</tr>
</tbody>
</table>

SEE ALSO PEXGSE(3)
NAME  PEXGSE-PEXSunGseIdSetStrokeJoin - PEX Set Stroke Join

SYNTAX  void PEXGSE(Display *display, XID resource_id, PEXOCRequestType req_type, long id, int length, char *data)

PARAMETERS  display  A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id  The resource identifier of the renderer or structure.
req_type  The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
id  The identifier of the GSE.
length  The length, in bytes, of the GSE data.
data  A pointer to the GSE data.

RETURNS  None

DESCRIPTION  To use the Sun SetStrokeJoin GSE, the id parameter should be set to PEXSunGSEIdSetStrokeJoin, the data parameter should be set to point to structure type pexSunGseStrokeJoin, and the length parameter should be set to the total length of data.
This GSE controls the appearance of wide-stroke joins.

DATA STRUCTURES  typedef struct {
   CARD32 prim;
   INT32 type;
} pexSunGseStrokeJoin;
The prim field specifies which primitives to apply the join-style to. Values for prim are defined in SunPEX.h as listed below:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEXSunJoinPrimLine</td>
<td>1</td>
</tr>
<tr>
<td>PEXSunJoinPrimEdge</td>
<td>4</td>
</tr>
</tbody>
</table>

The type field specifies the join-style. Values for type are defined in SunPEX.h as listed below:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEXSunJoinTypeMitred</td>
<td>0</td>
</tr>
<tr>
<td>PEXSunJoinTypeRound</td>
<td>1</td>
</tr>
<tr>
<td>PEXSunJoinTypeBeveled</td>
<td>2</td>
</tr>
<tr>
<td>PEXSunJoinTypeAny</td>
<td>3</td>
</tr>
</tbody>
</table>

SEE ALSO  PEXGSE(3)
## NAME
PEXGSE-PEXSunGseIdSetSurfTranspCoef - PEX Set Surface Transparency Coefficient

## SYNTAX
```c
void PEXGSE(Display *display, XID resource_id, PEXOCRequestType req_type, long id,
    int length, char *data)
```

## PARAMETERS
- **display**: A pointer to a display structure returned by a successful `XOpenDisplay` call.
- **resource_id**: The resource identifier of the renderer or structure.
- **req_type**: The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- **id**: The identifier of the GSE.
- **length**: The length, in bytes, of the GSE data.
- **data**: The data structure.

## RETURNS
None

## DESCRIPTION
To use the Sun SetSurfaceTransparencyCoefficient GSE, the `id` parameter should be set to `PEXSunGSEIdSetSurfTranspCoef`, the `data` parameter should be set to point to structure type `pexSunGseTranspCoef`, and the `length` parameter should be set to the total length of `data`.

This GSE sets a surface transparency coefficient to be applied when rendering subsequent surface primitives.

### DATA STRUCTURES
```c
typedef struct {
    PEXFLOAT value;
} pexSunGseTranspCoef;
```

The `value` field can range from 0.0 to 1.0.

## SEE ALSO
PEXGSE(3)
**NAME**
PEXGSE-PEXSunGseIdSetTextSlantAngle - PEX Set Text Slant Angle

**SYNTAX**
void PEXGSE(Display ∗display, XID resource_id, PEXOCRequestType req_type, long id, int length, char ∗data)

**PARAMETERS**
- *display* A pointer to a display structure returned by a successful XOpenDisplay call.
- *resource_id* The resource identifier of the renderer or structure.
- *req_type* The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- *id* The identifier of the GSE.
- *length* The length, in bytes, of the GSE data.
- *data* A pointer to the GSE data.

**RETURNS**
None

**DESCRIPTION**
To use the Sun SetTextSlantAngle GSE, the *id* parameter should be set to PEXSunGSEIdSetTextSlantAngle, the *data* parameter should be set to point to structure type pexSunGseTextSlantAngle, and the *length* parameter should be set to the total length of *data*. This GSE causes the vertical component of subsequent text output commands to be slanted from the character up vector by *slant_angle*. The slant angle, in radians, can take values between -p/2 and +p/2. A negative slant angle will slant the characters in the backward direction.

**DATA STRUCTURES**
typedef struct {
    PEXFLOAT slant_angle;
} pexSunGseTextSlantAngle;

**SEE ALSO**
PEXGSE(3)

modified November 1995
NAME

PEXGeoNormFillArea - utility function

SYNTAX

int PEXGeoNormFillArea(unsigned int facet_attributes, unsigned int vertex_attributes, int color_type, PEXFacetData *facet_data, unsigned int count, PEXArrayOfVertex vertices)

PARAMETERS

facet_attributes A mask indicating the facet attributes provided. It should contain the bit PEXGANormal.
vertex_attributes A mask indicating the vertex attributes provided.
color_type The type of color data provided.
facet_data A pointer to facet data. This function adds the geometric normal to this data.
count The number of vertices.
vertices An array of vertices defining the fill area.

RETURNS

None

DESCRIPTION

This function computes the geometric normal of a fill area and stores it in the specified facet data.

The normal is computed by finding the first three non-colinear points in the specified vertices, forming two vectors from those points, one from the first point to the second point and one from the first point to the third point, and computing the cross product of those two vectors. The geometric normal is the normalized cross product.

The three points, A, B, and C are selected as follows. Point A is the first point in the list of vertices. Point B is the next point in the list that is not coincident with A. Point C is the next point in the list that is not colinear with A and B. If it is not possible to find three such points, the functions returns unsuccessfully.

If the facet attributes does not contain the bit PEXGANormal, the geometric normal is not computed. However, the function still returns successfully.

ERRORS

None
NAME
PEXGeoNormFillAreaSet - utility function

SYNTAX
int PEXGeoNormFillAreaSet(unsigned int facet_attributes, unsigned int vertex_attributes, int color_type, unsigned int count, PEXFacetData *facet_data, PEXListOfVertex *vertex_lists)

PARAMETERS
facet_attributes A mask indicating the facet attributes provided. It should contain the bit PEXGANormal.
vertex_attributes A mask indicating the vertex attributes provided.
color_type The type of color data provided.
count The number of fill areas in the set.
facet_data An array of facet data. This function adds the geometric normal to this data.
vertex_lists A pointer to the list of vertex arrays defining each contour of the fill area set.

RETURNS
Zero if successful; otherwise, one of the following:
PEXBadPrimitive
A normal cannot be computed because all fill areas in the set are degenerate or because the vertices of all the fill areas are colinear.

DESCRIPTION
This function computes the geometric normal of a fill area set and stores it in the specified facet data.

The normal is computed by finding the first three non-colinear points in a fill area of the set, beginning with the first fill area and searching until three such points are found in a single fill area. Two vectors are formed from these points: one vector from the first point to the second point, and one vector from the first point to the third point. The geometric normal returned is the normalized cross product of those two vectors.

The three points, A, B, and C are selected as follows. Point A is the first point in the first list of vertices. Point B is the next point in that same list that is not coincident with A. Point C is the next point in that same list that is not colinear with A and B. If it is not possible to find three such points in the first list, then the rest of the lists are searched in order to select three appropriate points from a single list. If it is still not possible to find three such points in any list, the functions returns unsuccessfully.

If the facet attributes does not contain the bit PEXGANormal, the geometric normal is not computed. However, the function still returns successfully.

ERRORS
None
NAME
PEXGeoNormQuadrilateralMesh - utility function

SYNTAX
int PEXGeoNormQuadrilateralMesh(unsigned int facet_attributes, unsigned int vertex_attributes, int color_type, PEXArrayOfFacetData facet_data, unsigned int col_count, unsigned int row_count, PEXArrayOfVertex vertices)

PARAMETERS
- facet_attributes: A mask indicating the facet attributes provided. It should contain the bit PEXGANormal.
- vertex_attributes: A mask indicating the vertex attributes provided.
- color_type: The type of color data provided.
- facet_data: An array of facet data. This function adds the geometric normal to this data.
- col_count: The number of columns in the vertex array.
- row_count: The number of rows in the vertex array.
- vertices: A two-dimensional (row-major) array of vertices defining the quadrilateral mesh.

RETURNS
Zero if successful; otherwise, one of the following:
- PEXBadPrimitive
  A normal cannot be computed for one or more quadrilaterals in the mesh.

DESCRIPTION
This function computes the geometric normals of a quadrilateral mesh and stores them in the specified facet data.

The geometric normal of each quadrilateral is computed by forming two vectors from two of its sides, and computing the cross product of those two vectors. The geometric normal is the normalized cross product:

\[ \mathbf{N}_g = \frac{(\mathbf{V}_1 \times \mathbf{V}_2)}{|\mathbf{V}_1 \times \mathbf{V}_2|} \]

Given the quadrilateral composed of four vertices, \( P_{ij} \), where \( i \) indicates the row of the point and \( j \) its column, the first vector, \( \mathbf{V}_1 \), is from \( P_{ij} \) to \( P_{i+1,j+1} \). The second vector, \( \mathbf{V}_2 \), is from \( P_{i+1,j} \) to \( P_{i,j+1} \).

If the facet attributes does not contain the bit PEXGANormal, the geometric normal is not computed. However, the function still returns successfully.

A geometric normal is computed for all quadrilaterals where it is possible to compute one, even if a normal cannot be computed for some other quadrilaterals. An error is returned if a normal cannot be computed for one or more of the quadrilaterals in the mesh.

ERRORS
None
NAME  PEXGeoNormSetOfFillAreaSets - utility function

SYNTAX  int PEXGeoNormSetOfFillAreaSets(unsigned int facet_attributes, unsigned int vertex_attributes, int color_type, unsigned int set_count, PEXArrayOfFacetData facet_data, unsigned int vertex_count, PEXArrayOfVertex vertices, unsigned int index_count, PEXConnectivityData *connectivity)

PARAMETERS  

facet_attributes  A mask indicating the facet attributes provided. It should contain the bit PEXGANormal.

vertex_attributes  A mask indicating the vertex attributes provided.

color_type  The type of color data provided.

set_count  The number of fill area sets.

facet_data  An array of facet data. This function adds the geometric normals to this data.

vertex_count  The number of vertices.

vertices  An array of vertices.

index_count  The number of vertex connectivity indices.

connectivity  A pointer to the list of contour connectivity data.

RETURNS  Zero if successful; otherwise, one of the following:

PEXBadPrimitive  A geometric normal is computed for all fill area sets where it is possible to compute one, even if a normal cannot be computed for some other fill area sets. The function returns unsuccessfully if a normal cannot be computed for one or more of the fill area sets.
typedef struct {
    unsigned short count;  /* number of lists */
    PEXListOfUShort *lists;
} PEXConnectivityData;

typedef struct {
    unsigned short count;  /* number of shorts */
    unsigned short *shorts;
} PEXListOfUShort;

See also PEXlib.h.

ERRORS  None
### NAME
PEXGeoNormTriangleStrip - utility function

### SYNTAX
```c
int PEXGeoNormTriangleStrip(unsigned int facet_attributes, unsigned int vertex_attributes, int color_type, PEXArrayOfFacetData facet_data, unsigned int count, PEXArrayOfVertex vertices)
```

### PARAMETERS
- **facet_attributes**
  A mask indicating the facet attributes provided. It should contain the bit PEXGANormal.
- **vertex_attributes**
  A mask indicating the vertex attributes provided.
- **color_type**
  The type of color data provided.
- **facet_data**
  An array of facet data. This function adds the geometric normals to this data.
- **count**
  The number of vertices.
- **vertices**
  An array of vertices defining the triangle strip.

### RETURNS
Zero if successful; otherwise, one of the following:
- **PEXBadPrimitive**
  A normal cannot be computed for one or more triangles in the strip.

### DESCRIPTION
This function computes the geometric normals of a triangle strip and stores them in the specified facet data.

The geometric normal of each triangle is computed by forming two vectors from two of its sides, and computing the cross product of those two vectors. The geometric normal is the normalized cross product:

\[ \text{Ng} = \frac{(V1 \times V2)}{|V1 \times V2|} \]

For the first, third, and subsequent odd-numbered triangles, the first vector (V1) is from the first point (Pi) of the triangle to the second point (Pi+1), and the second vector (V2) is from the first point of the triangle to the third point (Pi+2). For the second, fourth, and subsequent even-numbered triangles, the first vector is from the first point (Pi) of the triangle to the third point (Pi+2), and the second vector is from the first point of the triangle to the second point (Pi+1).

If the facet attributes does not contain the bit PEXGANormal, the geometric normal is not computed. However, the function still returns successfully.

A geometric normal is computed for all triangles where it is possible to compute one, even if a normal cannot be computed for some other triangles. An error is returned if a normal cannot be computed for one or more of the triangles in the strip.

### ERRORS
None

---

modified November 1995

163
NAME  PEXGetAncestors - Get Ancestors

SYNTAX  PEXStructurePath *PEXGetAncestors(Display *display, PEXStructure structure, int path_part, unsigned long path_depth, unsigned long *count_return)

PARAMETERS  display A pointer to a display structure returned by a successful XOpenDisplay call.
structure The resource identifier of the structure.
path_part The part of the path to return (PEXTopPart or PEXBottomPart).
path_depth The maximum number of structure network path levels to be returned in each path found.
count_return Returns the number of paths found.

RETURNS An array of structure paths defining the ancestors of the specified structure; a null pointer if unsuccessful.

DESCRIPTION This function returns a list of structure network paths which reference the specified structure. Paths are returned as lists of element references, each of which is represented as a structure resource identifier and an offset that gives the element’s position in the structure. Only unique paths are returned; in other words there will be no duplicates in the list of returned paths.

The path part must be either PEXTopPart, which requests that the top of the structure paths be returned, or PEXBottomPart, which requests that the bottom of the structure paths be returned.

The path depth specifies the maximum number of element references to be returned in each path. If the path depth is 0, the entire path is returned. If the path depth is 1, only one element reference is returned for each path. A path depth of 2 returns two elements, and so on.

Specifying a path depth of 0 and a path part of PEXTopPart returns the unique top parts of all paths to structure. Specifying a path depth of 1 and a path part of PEXTopPart returns the root structure of all structure networks which contain structure. A path depth of 2 and path part of PEXBottomPart returns all of the structure's immediate ancestors. Determine the number of references to structure by setting the path depth to 1 and the path part to PEXBottomPart.

PEXlib allocates memory for the returned ancestor information. PEXFreeStructurePaths(3) should be called to deallocate the memory.

DATA STRUCTURES typedef XID PEXStructure;
typedef struct {
    unsigned long count; /* number of elements */
    PEXElementRef *elements;
} PEXStructurePath;
```c
typedef struct {
    PEXStructure structure;
    unsigned long offset;
} PEXElementRef;
```

**ERRORS**

**BadPEXStructure**
- The specified `structure` resource identifier is invalid.

**BadValue**
- The specified value for path part is invalid.

**SEE ALSO**
- `PEXCreateStructure(3)`
- `PEXGetStructuresInNetwork(3)`
- `PEXGetDescendants(3)`
NAME  PEXGetDefinedIndices - Get Lookup Table Defined Indices

SYNTAX  Status PEXGetDefinedIndices(Display *display, PEXLookupTable table, unsigned long *count_return, PEXTableIndex **indices_return)

PARAMETERS  display  A pointer to a display structure returned by a successful XOpenDisplay call.
  table  The resource identifier of the lookup table.
  count_return  Returns the number of returned indices.
  indices_return  Returns an array of indices to defined table entries.

RETURNS  Zero if unsuccessful, non-zero otherwise.

DESCRIPTION  This function returns the defined indices for the specified lookup table. The index of each defined table entry is returned in a list. PEXlib allocates memory for the returned indices. XFree should be called to deallocate the memory.

DATA STRUCTURES  typedef XID PEXLookupTable;
  typedef unsigned short PEXTableIndex;

ERRORS  BadPEXLookupTable
  The specified lookup table resource identifier is invalid, or the table type is unsupported.

SEE ALSO  PEXCreateLookupTable(3)
  PEXGetTableInfo(3)
NAME  PEXGetDescendants - Get Descendants

SYNTAX  PEXStructurePath *PEXGetDescendants(Display *display, PEXStructure structure, int path_part, unsigned long path_depth, unsigned long *count_return)

PARAMETERS  
- display  A pointer to a display structure returned by a successful XOpenDisplay call.
- structure  The resource identifier of the structure.
- path_part  The part of the path to return (PEXTopPart or PEXBottomPart).
- path_depth  The maximum number of structure network path levels to be returned in each path found.
- count_return  Returns the number of paths found.

DESCRIPTION  
This function returns a list of structure network paths referenced by the specified structure. The elements of the returned array are of type PEXStructurePath. Paths are returned as lists of element references, each of which is represented as a structure resource identifier and an offset that gives the element’s position in the structure. Only unique paths are returned; in other words, there will be no duplicates in the returned paths.

The path part must be either PEXTopPart, which requests that the top of the structure paths be returned, or PEXBottomPart, which requests that the bottom of the structure paths be returned.

The path depth specifies the maximum number of element references to be returned in each path. If the path depth is 0, the entire path is returned. If the path depth is 1, only one element reference is returned for each path, and so on.

For instance, specifying a path depth of 0 and a path part of PEXTopPart returns all paths from structure to leaf nodes in the structure network tree in the order they would be traversed. Specifying a path depth of 1 and a path part of PEXBottomPart determines the bottom-most structures of the structure network rooted at structure.

PEXlib allocates memory for the returned descendant information. PEXFreeStructurePaths(3) should be called to deallocate the memory.

DATA STRUCTURES  
typedef XID PEXStructure;

typedef struct {
  unsigned long count; /* number of elements */
  PEXElementRef *elements;
} PEXStructurePath;

typedef struct {
  PEXStructure structure;
  unsigned long offset;
} PEXElementRef;

modified November 1995
### ERRORS

- **BadPEXStructure**
  The specified structure resource identifier is invalid.
- **BadValue**
  The specified value for path part is invalid.

### SEE ALSO

- PEXCreateStructure(3)
- PEXGetStructuresInNetwork(3)
- PEXGetAncestors(3)
NAME
PEXGetElementInfo - Get Element Information

SYNTAX
Status PEXGetElementInfo(Display *display, PEXStructure structure, int whence1, long offset1, int whence2, long offset2, int float_format, unsigned long *count_return, PEXElementInfo **info_return)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
structure The resource identifier of the structure.
whence1 A value specifying, with offset1, the first limit of the range of queried elements (PEXBeginning, PEXCurrent, PEXEnd).
offset1 The offset from whence1 denoting the first limit of the range of queried elements.
whence2 A value specifying, with offset2, the second limit of the range of elements to be queried (PEXBeginning, PEXCurrent, PEXEnd).
offset2 The offset from whence2 denoting the second limit of the range of elements to be queried.
float_format The floating point format to use when computing element sizes (PEXIEEE_754_32, PEXDEC_F_Floating, PEXIEEE_754_64, PEXDEC_D_Floating).
count_return Returns the number element info records returned.
info_return Returns an array of element info records describing the elements in the specified range.

RETURNS
Zero if unsuccessful, non-zero otherwise.

DESCRIPTION
This function returns information about a range of elements from the specified structure. If a computed offset is less than zero it is set to zero before obtaining the element information. If a computed offset is greater than the number of elements in the structure, it is set to the offset of the last structure element in the structure. The element pointer attribute of structure is not affected by this command.

Information returned about the list of inquired elements includes the type of each element and its size. The size of each element is based upon the specified floating point format. No information is returned for inquires on element offset zero. The element pointer is not affected by this function.

PEXlib allocates memory for the return value. XFree should be called to deallocate the memory.
**DATA STRUCTURES**

typedef XID PEXStructure;

typedef struct {
    unsigned short type;
    unsigned short length;
} PEXElementInfo;

**ERRORS**

- **BadPEXFloatingPointFormat**
  The specified floating point format is invalid or unsupported.

- **BadPEXStructure**
  The specified *structure* resource identifier is invalid.

- **BadValue**
  The specified value for *whence* parameter is invalid.

**SEE ALSO**

- PEXCreateStructure(3)
### NAME
PEXGetEnumTypeInfo - Get Enumerated Type Information

### SYNTAX
```
Status PEXGetEnumTypeInfo(Display *display, Drawable drawable, unsigned long count,
                          int *enum_types, unsigned long item_mask, unsigned long **info_count_return,
                          PEXEnumTypeDesc **enum_info_return)
```

### PARAMETERS
- **display**: A pointer to a display structure returned by a successful `XOpenDisplay` call.
- **drawable**: The resource identifier of a drawable.
- **count**: The number of enumerated types.
- **enum_types**: A list of enumerated types for which information is to be returned. (See the Description for valid values.)
- **item_mask**: A mask indicating the data to be returned for each enumerated type. (See the Description for valid values.)
- **info_count_return**: Returns an array of counts. For each enumerated type, there is an entry specifying the number of descriptors in the return value array.
- **enum_info_return**: Returns an array of enumerated type descriptors containing the enumerated type information.

### DESCRIPTION
PEXGetEnumTypeInfo allows the application to inquire the supported values for each enumerated type. It returns a descriptor for each supported value of each specified enumerated type requested. These values will be valid for all drawables having the same root window and depth as the specified drawable.

The standard PEX enumerated types are:

- `PEXETATextStyle`
- `PEXETColorApproxModel`
- `PEXETColorApproxType`
- `PEXETColorType`
- `PEXETCurveApproxMethod`
- `PEXETDisplayUpdateMode`
- `PEXETEscape`
- `PEXETFloatFormat`
- `PEXETGDP2D`
- `PEXETGDP`  
- `PEXETGSE`
- `PEXETHatchStyle`
- `PEXETHLSRMode`
- `PEXETInteriorStyle`
- `PEXETLightType`
- `PEXETMarkerType`
- `PEXETModelClipOperator`
- `PEXETParaSurfCharacteristics`
- `PEXETPickAllMethod`
- `PEXETPickDeviceType`
- `PEXETPickOneMethod`
- `PEXETPolylineInterpMethod`
- `PEXETPromptEchoType`
- `PEXETReflectionModel`
- `PEXETRenderingColorModel`
- `PEXETSURFACETEXCOORD0`  
- `PEXETSURFACETEXCOORD1`
- `PEXETSURFACEINTERPOLATION`
- `PEXETSURFACEAPPROXMETHOD`
- `PEXETSURFACEAPPROXOPERATOR`
- `PEXETTrimCurveApproxMethod`

modified November 1995
Enumerated type descriptors are made up of an index field and a mnemonic field. The index field contains the numeric value of the supported type, and the mnemonic field contains a string describing the type.

The item mask specifies which of the fields is returned in each enumerated type descriptor. Defined values for the item mask are `PEXETCounts`, `PEXETIndex`, `PEXETMnemonic` and `PEXETAll`. If the item mask is `PEXETCounts`, no descriptor values are returned; only the counts will be returned. If the item mask is `PEXETIndex`, the supported index values are returned in addition to the counts. If the item mask is `PEXETMnemonic`, the mnemonic strings describing the supported values are returned in addition to the counts. If the item mask is `PEXETAll`, both the supported index values and the mnemonic strings are returned in addition to the counts.

PEXlib allocates memory for the returned array of counts and for the return value array of enumerated type descriptors. `PEXFreeEnumInfo(3)` should be called to deallocate this memory.

The following are the standard enumerated type index, mnemonic pairs which may be returned.

**PEXETATextStyle**
- `PEXATextNotConnected`, `PEXETMTextNotConnected`
- `PEXATextConnected`, `PEXETMTextConnected`

**PEXETColorApproxModel**
- `PEXColorApproxRGB`, `PEXETMColorApproxRGB`
- `PEXColorApproxCIE`, `PEXETMColorApproxCIE`
- `PEXColorApproxHSV`, `PEXETMColorApproxHSV`
- `PEXColorApproxHLS`, `PEXETMColorApproxHLS`
- `PEXColorApproxYIQ`, `PEXETMColorApproxYIQ`

**PEXETColorApproxType**
- `PEXColorSpace`, `PEXETMColorSpace`
- `PEXColorRange`, `PEXETMColorRange`

**PEXETColorType**
- `PEXColorTypeIndexed`, `PEXETMColorTypeIndexed`
- `PEXColorTypeRGB`, `PEXETMColorTypeRGB`
- `PEXColorTypeCIE`, `PEXETMColorTypeCIE`
- `PEXColorTypeHSV`, `PEXETMColorTypeHSV`
- `PEXColorTypeHLS`, `PEXETMColorTypeHLS`
- `PEXColorTypeRGB8`, `PEXETMColorTypeRGB8`
- `PEXColorTypeRGB16`, `PEXETMColorTypeRGB16`

**PEXETCurveApproxMethod**
- `PEXApproxImpDep`, implementation-dependent string
- `PEXApproxConstantBetweenKnots`, `PEXETMApproxConstantBetweenKnots`
- `PEXApproxWCCChordalSize`, `PEXETMApproxWCCChordalSize`
- `PEXApproxNPCChordalSize`, `PEXETMApproxNPCChordalSize`
PEXApproxDCChordalSize, PEXETMAproxDCChordalSize
PEXCurveApproxWCChordalDev, PEXETMCurveApproxWCChordalDev
PEXCurveApproxNPCChordalDev, PEXETMCurveApproxNPCChordalDev
PEXCurveApproxDCChordalDev, PEXETMCurveApproxDCChordalDev
PEXApproxWCRelative, PEXETMAproxWCRelative
PEXApproxNPCRelative, PEXETMAproxNPCRelative
PEXApproxDCRelative, PEXETMAproxDCRelative

PEXETDisplayUpdateMode
  PEXVisualizeEach, PEXETMVisualizeEach
  PEXVisualizeEasy, PEXETMVisualizeEasy
  PEXVisualizeNone, PEXETMVisualizeNone
  PEXSimulateSome, PEXETMSimulateSome
  PEXVisualizeWhenever, PEXETMVisualizeWhenever

PEXETEscape
  PEXEscapeSetEchoColor, PEXETMEscapeSetEchoColor

PEXETFloatFormat
  PEXIEEE_754_32, PEXETMIEEE_754_32
  PEXDEC_F_Floating, PEXETMDEC_F_Floating
  PEXIEEE_754_64, PEXETMIEEE_754_64
  PEXDEC_D_Floating, PEXETMDEC_D_Floating

PEXETGDP2D
  There are currently no standard 2D generalized drawing primitives.

PEXETGDP
  There are currently no standard 3D generalized drawing primitives.

PEXETGSE
  There are currently no standard generalized structure elements.

PEXETHatchStyle
  There are currently no standard hatch styles.

PEXETHLHSRMode
  PEXHLHSROff, PEXETMHLHSROff
  PEXHLHSRZBuffer, PEXETMHLHSRZBuffer
  PEXHLHSRPainters, PEXETMHLHSRPainters
  PEXHLHSRScanline, PEXETMHLHSRScanline
  PEXHLHSRHiddenLineOnly, PEXETMHLHSRHiddenLineOnly
  PEXHLHSRZBufferID, PEXETMHLHSRZBufferID

PEXETInteriorStyle
  PEXInteriorStyleHollow, PEXETMInteriorStyleHollow
  PEXInteriorStyleSolid, PEXETMInteriorStyleSolid
  PEXInteriorStylePattern, PEXETMInteriorStylePattern
  PEXInteriorStyleHatch, PEXETMInteriorStyleHatch

modified November 1995
PEXGetEnumTypeInfo (3)

PEXInteriorStyleEmpty, PEXETMInteriorStyleEmpty

PEXETLightType
PEXLightAmbient, PEXETMLightAmbient
PEXLightWCVector, PEXETMLightWCVector
PEXLightWCPoint, PEXETMLightWCPoint
PEXLightWCSpot, PEXETMLightWCSpot

PEXETLineType
PEXLineTypeSolid, PEXETMLineTypeSolid
PEXLineTypeDashed, PEXETMLineTypeDashed
PEXLineTypeDotted, PEXETMLineTypeDotted
PEXLineTypeDashDot, PEXETMLineTypeDashDot

PEXETMarkerType
PEXMarkerDot, PEXETMMarkerDot
PEXMarkerCross, PEXETMMarkerCross
PEXMarkerAsterisk, PEXETMMarkerAsterisk
PEXMarkerCircle, PEXETMMarkerCircle
PEXMarkerX, PEXETMMarkerX

PEXETModelClipOperator
PEXModelClipReplace, PEXETMModelClipReplace
PEXModelClipIntersection, PEXETMModelClipIntersection

PEXETParaSurfCharacteristics
PEXPSCNone, PEXETMPSCNone
PEXPSCImpDep, implementation-dependent string
PEXPSCIsocurves, PEXETMPSCIsocurves
PEXPSCMCLevelCurves, PEXETMPSCMCLevelCurves
PEXPCWCLevelCurves, PEXETMPSCWCLevelCurves

PEXETPickAllMethod
PEXPickAllAll, PEXETMPickAllAll
PEXPickAllVisible, PEXETMPickAllVisible

PEXETPickDeviceType
PEXPickDeviceDCHitBox, PEXETMPickDeviceDCHitBox
PEXPickDeviceNPCHitVolume, PEXETMPickDeviceNPCHitVolume

PEXETPickOneMethod
PEXPickLast, PEXETMPickLast
PEXPickClosestZ, PEXETMPickClosestZ
PEXPickVisibleAny, PEXETMPickVisibleAny
PEXPickVisibleClosest, PEXETMPickVisibleClosest

modified November 1995
PEXETPolylineInterpMethod
  PEXPolylineInterpNone, PEXETMPolylineInterpNone
  PEXPolylineInterpColor, PEXETMPolylineInterpColor

PEXETPromptEchoType
  PEXEchoPrimitive, PEXETMEchoPrimitive
  PEXEchoStructure, PEXETMEchoStructure
  PEXEchoNetwork, PEXETMEchoNetwork

PEXETReflectionModel
  PEXReflectionNone, PEXETMReflectionNone
  PEXReflectionAmbient, PEXETMReflectionAmbient
  PEXReflectionDiffuse, PEXETMReflectionDiffuse
  PEXReflectionSpecular, PEXETMReflectionSpecular

PEXETRenderingColorModel
  PEXRenderingColorModelImpDep, implementation-dependent string
  PEXRenderingColorModelRGB, PEXETMRenderingColorModelRGB
  PEXRenderingColorModelCIE, PEXETMRenderingColorModelCIE
  PEXRenderingColorModelHSV, PEXETMRenderingColorModelHSV
  PEXRenderingColorModelHLS, PEXETMRenderingColorModelHLS

PEXETSurfaceApproxMethod
  PEXApproxImpDep, implementation-dependent string
  PEXApproxConstantBetweenKnots, PEXETMApproxConstantBetweenKnots
  PEXApproxWCChordalSize, PEXETMApproxWCChordalSize
  PEXApproxNPCChordalSize, PEXETMApproxNPCChordalSize
  PEXApproxDCChordalSize, PEXETMApproxDCChordalSize
  PEXSurfaceApproxWCPlanarDev, PEXETMSurfaceApproxWCPlanarDev
  PEXSurfaceApproxNPCPlanarDev, PEXETMSurfaceApproxNPCPlanarDev
  PEXSurfaceApproxDCPlanarDev, PEXETMSurfaceApproxDCPlanarDev
  PEXApproxWCRelative, PEXETMApproxWCRelative
  PEXApproxNPCRelative, PEXETMApproxNPCRelative
  PEXApproxDRelative, PEXETMApproxDRelative

PEXETSurfaceEdgeType
  PEXSurfaceEdgeSolid, PEXETMSurfaceEdgeSolid
  PEXSurfaceEdgeDashed, PEXETMSurfaceEdgeDashed
  PEXSurfaceEdgeDotted, PEXETMSurfaceEdgeDotted
  PEXSurfaceEdgeDashDot, PEXETMSurfaceEdgeDashDot

PEXETSurfaceInterpMethod
  PEXSurfaceInterpNone, PEXETMSurfaceInterpNone
  PEXSurfaceInterpColor, PEXETMSurfaceInterpColor
  PEXSurfaceInterpDotProduct, PEXETMSurfaceInterpDotProduct
  PEXSurfaceInterpNormal, PEXETMSurfaceInterpNormal
PEXETTrimCurveApproxMethod
  PEXApproxImpDep, implementation-dependent string
  PEXApproxConstantBetweenKnots, PEXETMApproxConstantBetweenKnots

DATA STRUCTURES
typedef struct {
  PEXEnumTypeIndex index;
  char *descriptor; /* null terminated string */
} PEXEnumTypeDesc;

ERRORS
  BadDrawable
    The specified drawable resource identifier is invalid.
  BadMatch
    The specified drawable is unsupported.
  BadValue
    A specified enumerated type is invalid.

SEE ALSO
  PEXFreeEnumInfo(3)
NAME  
PEXGetExtensionInfo - Get Extension Information

SYNTAX  
PEXExtensionInfo *PEXGetExtensionInfo(Display *display)

PARAMETERS  
display  
A pointer to a display structure returned by a successful XOpenDisplay call.

RETURNS  
A pointer to the extension information; a null pointer if unsuccessful.

DESCRIPTION  
PEXGetExtensionInfo allows an application program to inquire the extension information from a PEX server extension.

The major version number, minor version number, release number, vendor name, and subset information are returned. If the subset value is PEXCompleteImplementation, the extension is a full PEX implementation. If the subset value is PEXImmediateMode, the extension supports only the immediate rendering subset of PEX. If the subset value is PEXStructureMode, the extension supports only the structure rendering subset of PEX. If the subset value is (PEXImmediateMode and PEXStructureMode), the extension supports both the immediate rendering and the structure rendering subsets of PEX. If the subset value is PEXWorkstationOnly, the extension supports only the PHIGS workstation subset of PEX.

The memory returned by this function is private to PEXlib and must not be modified or freed by the application.

DATA STRUCTURES  
typedef struct {  
    unsigned short major_version;  
    unsigned short minor_version;  
    unsigned long release;  
    unsigned long subset_info;  
    char *vendor_name;  
    int major_opcode;  
    int first_event;  
    int first_error;  
} PEXExtensionInfo;

ERRORS  
None
NAME
PEXGetImpDepConstants - Get Implementation Dependent Constants

SYNTAX
Status PEXGetImpDepConstants(Display *display, Drawable drawable, unsigned long count, unsigned short *names, PEXImpDepConstant **constants_return)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
drawable The resource identifier of a drawable.
count The number of implementation-dependent constants.
names An array of names of implementation-dependent constants to be returned. (See the Description for valid values.)

constants_return Returns an array of implementation-dependent constants.

RETURNS
Zero if unsuccessful; non-zero otherwise.

DESCRIPTION
PEXGetImpDepConstants allows an application program to query one or more implementation-dependent constants.

A single integer or floating-point value is returned for each value requested. These values are returned in order, with one return value in constants for each item in names. The implementation-dependent constants returned are based on the values used for a drawable having the same root and depth as the drawable specified. PEXlib allocates memory for the returned constants. XFree should be called to deallocate the memory.

The following implementation-dependent constants are standard:

PEXIDBestColorApprox (integer) — Either PEXColorApproxPowersOf2 or PEXColorApproxAnyValues, depending on whether there is a significant performance gain if the number of reds/greens/blues in the color approximation table entry is a power of two, so pixels can be composed using shifts and adds.

PEXIDDitheringSupported (integer) — Either True if the dithering hint in color approximation lookup tables is used to control dithering, or False, if the dithering hint in color approximation lookup tables is not used.

PEXIDDoubleneBufferingSupported (integer) — Either True if the server supports double-buffering for workstation resources, or False, if the server does not supports double-buffering for workstation resources.

PEXIDMaxEdgeWidth (integer) — Width (in pixels) of widest edge that can be drawn.

PEXIDMaxHitsEventSupported (integer) — True if the server supports the PEXMaxHitsReached event, or False, if the server does not support the event.

PEXIDMaxLineWidth (integer) — Width (in pixels) of widest line or curve that can be drawn.

PEXIDMaxMarkerSize (integer) — Largest dimension (either height or width, in pixels) of largest marker that can be drawn. This maximum is exclusive of the marker type PEXMarkerDot which is always drawn as the smallest displayable point.

modified November 1995
PEXIDMaxModelClipPlanes (integer) — Maximum number of modeling clipping planes may be defined.

PEXIDMaxNameSetNames (integer) — Maximum number of names allowed in a name set.

PEXIDMaxNonAmbientLights (integer) — Maximum number of non-ambient light sources that can be enabled at one time.

PEXIDMaxNURBOrder (integer) — Maximum non-uniform rational B-spline order supported.

PEXIDMaxTrimCurveOrder (integer) — Maximum order for trim curves.

PEXIDMinEdgeWidth (integer) — Width (in pixels) of thinnest edge that can be drawn.

PEXIDMinLineWidth (integer) — Width (in pixels) of thinnest line or curve that can be drawn.

PEXIDMinMarkerSize (integer) — Largest dimension (either height or width, in pixels) of smallest marker that can be drawn. This minimum is exclusive of the marker type PEXMarkerDot which is always drawn as the smallest displayable point.

PEXIDNominalEdgeWidth (integer) — Width (in pixels) of standard edge.

PEXIDNominalLineWidth (integer) — Width (in pixels) of standard line or curve.

PEXIDNominalMarkerSize (integer) — Largest dimension (either height or width, in pixels) of standard marker.

PEXIDNumSupportedEdgeWidths (integer) — Number of supported edge widths. A value of 0 indicates that all edge widths, including fractional widths, between the minimum and maximum edge width are supported.

PEXIDNumSupportedLineWidths (integer) — Number of supported line or curve widths. A value of 0 indicates that all line widths, including fractional widths, between the minimum and maximum line width are supported.

PEXIDNumSupportedMarkerSizes (integer) — Number of supported marker sizes. A value of 0 indicates that all marker sizes, including fractional values, between the minimum and maximum marker size are supported.

PEXIDTransparencySupported (integer) — Either True if the transmission coefficient is utilized in the reflectance calculations, or False, if the transmission coefficient is not utilized.

PEXIDChromaticityRedU (flt_point) — CIEYUV u chromaticity coefficient for the red channel of the (properly adjusted) display device.

PEXIDChromaticityRedV (flt_point) — CIEYUV v chromaticity coefficient for the red channel of the (properly adjusted) display device.

PEXILuminanceRed (flt_point) — CIEYUV luminance value for the red channel of the (properly adjusted) display device.

PEXIDChromaticityGreenU (flt_point) — CIEYUV u chromaticity coefficient for the green channel of the (properly adjusted) display device.
PEXIDChromaticityGreenV (flt_point) — CIEYUV v chromaticity coefficient for the green channel of the (properly adjusted) display device.

PEXIDLuminanceGreen (flt_point) — CIEYUV luminance value for the green channel of the (properly adjusted) display device.

PEXIDChromaticityBlueU (flt_point) — CIEYUV u chromaticity coefficient for the blue channel of the (properly adjusted) display device.

PEXIDChromaticityBlueV (flt_point) — CIEYUV v chromaticity coefficient for the blue channel of the (properly adjusted) display device.

PEXIDLuminanceBlue (flt_point) — CIEYUV luminance value for the blue channel of the (properly adjusted) display device.

PEXIDChromaticityWhiteU (flt_point) — CIEYUV u chromaticity coefficient for the reference white of the (properly adjusted) display device.

PEXIDChromaticityWhiteV (flt_point) — CIEYUV v chromaticity coefficient for the reference white of the (properly adjusted) display device.

PEXIDLuminanceWhite (flt_point) — CIEYUV luminance value for the reference white of the (properly adjusted) display device.

DATA STRUCTURES
typedef union {
    unsigned long integer;
    float flt_point;
} PEXImpDepConstant;

ERRORS
BadDrawable
The specified drawable resource identifier is invalid.

BadMatch
The specified drawable is unsupported.

BadValue
A specified implementation-dependent constant name is invalid.
NAME
PEXGetNameSet - Get Name Set

SYNTAX
Status PEXGetNameSet(Display *display, PEXNameSet nameset, unsigned long *
count_return, PEXName **names_return)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay
call.
nameset The resource identifier of the name set to be queried.
count_return Returns the number of names.
names_return Returns an array of names.

RETURNS
Zero if unsuccessful; non-zero otherwise.

DESCRIPTION
This function returns the names in the specified name set. PEXlib allocates memory for the
return value. XFree should be called to deallocate the memory.

DATA STRUCTURES
typedef XID PEXNameSet;
typedef unsigned long PEXName;

ERRORS
BadPEXNameSet
The specified name set resource identifier is invalid.

SEE ALSO
PEXCreateNameSet(3)
PEXChangeNameSet(3)

modified November 1995
NAME        PEXGetOCAddr - Get Address For Encoded Output Commands

SYNTAX      char *PEXGetOCAddr(Display *display, int length)

PARAMETERS  

display     A pointer to a display structure returned by a successful XOpenDisplay call.

length      The number of bytes of data to be written by the application.

RETURNS     A pointer to memory where the application can write output command data; a null pointer if unsuccessful.

DESCRIPTION This function returns a memory address to the specified number of bytes in the transport buffer where the application can write data.

The pointer returned is valid only until the next PEXGetOCAddr or PEXCopyBytesToOC(3) is called.

An attempt to request more bytes than remaining in the transport buffer, or more bytes than returned by PEXGetOCAddrMaxSize(3), will result in an unsuccessful return value.

PEXStartOCs(3) must be called prior to this.

DO NOT attempt to deallocate or free memory at the address returned by this function.

ERRORS      None

SEE ALSO    PEXStartOCs(3)
             PEXFinishOCs(3)
             PEXCopyBytesToOC(3)
             PEXGetOCAddrMaxSize(3)
<table>
<thead>
<tr>
<th><strong>NAME</strong></th>
<th>PEXGetOCAddrMaxSize - Macro to Obtain the Maximum Size for PEXGetOCAddr</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SYNTAX</strong></td>
<td>PEXGetOCAddrMaxSize(display)</td>
</tr>
<tr>
<td><strong>PARAMETERS</strong></td>
<td>display A pointer to a display structure returned by a successful XOpenDisplay call.</td>
</tr>
<tr>
<td><strong>DESCRIPTION</strong></td>
<td>This macro evaluates to the maximum size for the length parameter of PEXGetOCAddr(3).</td>
</tr>
<tr>
<td><strong>ERRORS</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>SEE ALSO</strong></td>
<td>PEXGetOCAddr(3)</td>
</tr>
</tbody>
</table>

modified November 1995
NAME
PEXGetPickDevice - Get Pick Device Attributes

SYNTAX
PEXPDAttributes *PEXGetPickDevice(Display *display, PEXWorkstation workstation, int pick_device_type, unsigned long value_mask)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
workstation The resource identifier of the workstation.
pick_device_type The pick device type (PEXPickDeviceDCHitBox or PEXPickDeviceNPHitVolume).
value_mask A mask indicating which attributes to return.

RETURNS A pointer to the pick device attribute values; a null pointer if unsuccessful.

DESCRIPTION
This function returns the attribute values of a pick descriptor for the PHIGS workstation resource specified. The descriptor returned will be the currently-defined descriptor for the pick device of the type specified. Supported pick device types are inquirable via PEXGetEnumTypeInfo(3). The value mask indicates which attributes are to be returned. The value mask is constructed by OR’ing together the following constants:

PEXPDEchoSwitch
PEXPDEchoVolume
PEXPDPickDataRec
PEXPDPickExcl
PEXPDPickIncl
PEXPDPickPath
PEXPDPickPathOrder
PEXPDPickStatus
PEXPDPromptEchoType

PEXlib allocates memory for the return value. PEXFreePDAttributes(3) should be called to deallocate the memory.

DATA STRUCTURES
typedef XID PEXWorkstation;
typedef struct {
  unsigned short status;
  PEXPickPath path;
  int path_order;
  PEXNameSet inclusion;
  PEXNameSet exclusion;
  PEXPickRecord pick_record;
  PEXEnumTypeIndex prompt_echo_type;
  PEXViewport echo_volume;
  int echo_switch;
} PEXPDAttributes;
typedef struct {
    unsigned long count; /* number of elements */
    PEXPickElementRef *elements;
} PEXPickPath;

typedef struct {
    PEXStructure sid;
    unsigned long offset;
    unsigned long pick_id;
} PEXPickElementRef;

typedef XID PEXStructure;

typedef XID PEXNameSet;

typedef union {
    PEXPDNPCSubVolume volume;
    PEXPDDCHitBox box;
    PEXPickDataRecord data;
} PEXPickRecord;

typedef PEXNPCSubVolume PEXPDNPCHitVolume;

typedef struct {
    PEXCoord min;
    PEXCoord max;
} PEXNPCSubVolume;

typedef struct {
    float x;
    float y;
    float z;
} PEXCoord;

typedef struct {
    PEXDeviceCoord2D position;
    float distance;
} PEXPDDCHitBox;

typedef struct {
    short x;
    short y;
} PEXDeviceCoord2D;
typedef struct {
    unsigned short length; /* number of bytes in record */
    char *record;
} PEXPickDataRecord;

typedef short PEXEnumTypeIndex;

typedef struct {
    PEXDeviceCoord min;
    PEXDeviceCoord max;
    PEXSwitch use_drawable;
    unsigned char reserved[3];
} PEXViewport;

typedef struct {
    short x;
    short y;
    float z;
} PEXDeviceCoord;

typedef unsigned char PEXSwitch;

ERRORS
BadPEXWorkstation
   The specified workstation resource identifier is invalid.

BadValue
   The specified pick device type is invalid, or an invalid bit set in the value mask.

SEE ALSO
PEXChangePickDevice(3)
PEXGetEnumTypeInfo(3)
NAME
PEXGetPickMeasure - Get Pick Measure Attributes

SYNTAX
PEXPMAtributes *PEXGetPickMeasure(Display *display, PEXPickMeasure pick_measure, unsigned long value_mask)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
pick_measure The resource identifier of the pick measure.
value_mask A mask indicating which attributes to return.

RETURNS
A pointer to the pick measure attribute values; a null pointer if unsuccessful.

DESCRIPTION
This function returns the specified pick measure attribute values. The value mask indicates which attributes are to be returned. The value mask is constructed by OR'ing together the following constants:

PEXPMStatus
PEXPMPath

PEXlib allocates memory for the return value. PEXFreePMAttributes(3) should be called to deallocate the memory.

DATA STRUCTURES
typedef XID PEXPickMeasure;

typedef struct {
    unsigned short status;
    PEXPickPath pick_path;
} PEXPMAttributes;

typedef struct {
    unsigned long count; /* number of elements */
    PEXPickElementRef *elements;
} PEXPickPath;

typedef struct {
    PEXStructure sid;
    unsigned long offset;
    unsigned long pick_id;
} PEXPickElementRef;

typedef XID PEXStructure;

ERRORS
BadPEXPickMeasure
The specified pick_measure resource identifier is invalid.

BadValue
An invalid bit is set in the value mask.

modified November 1995
SEE ALSO

PEXCreatePickMeasure(3)
NAME
PEXGetPipelineContext - Get Pipeline Context Attributes

SYNTAX
PEXPCAttributes *PEXGetPipelineContext(Display *display, PEXPipelineContext context, unsigned long *value_mask)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
context The resource identifier of the pipeline context.
value_mask A pointer to an array of three unsigned long.

RETURNS
A pointer to pipeline context values; a null pointer if unsuccessful.

DESCRIPTION
This function returns the requested attribute values of the pipeline context. The value mask indicates which attribute values are requested. PEXSetPCAttributeMask and PEXSetPCAttributeMaskAll can be called to setup the value mask. PEXlib allocates the memory for the return value. PEXFreePCAttributes(3) should be called to deallocate the memory.

DATA STRUCTURES
typedef XID PEXPipelineContext;
See also PEXlib.h.

ERRORS
BadPEXPipelineContext
The specified pipeline context resource identifier is invalid.
BadValue
An invalid bit is set in the value mask.

SEE ALSO
PEXCreatePipelineContext(3)
PEXChangePipelineContext(3)
NAME  PEXGetPredefinedEntries - Get Lookup Table Predefined Indices

SYNTAX  Status PEXGetPredefinedEntries(Display *display, Drawable drawable, int table_type,
                                           unsigned int start, unsigned int count, PEXPointer *entries_return)

PARAMETERS

  display  A pointer to a display structure returned by a successful XOpenDisplay call.

  drawable  The resource identifier of a drawable.

  table_type  The type of lookup table (see the Description).

  start  The index of the first predefined entry to be returned.

  count  The number of predefined entries requested.

  entries_return  Returns an array of predefined table entries.

RETURNS  Zero if unsuccessful, non-zero otherwise.

DESCRIPTION  This function returns the predefined entries for the specified lookup table. The type must be one of the following:

  PEXLUTColor
  PEXLUTColorApprox
  PEXLUTDepthCue
  PEXLUTEdgeBundle
  PEXLUTInteriorBundle
  PEXLUTLight
  PEXLUTLineBundle
  PEXLUTMarkerBundle
  PEXLUTPattern
  PEXLUTTextBundle
  PEXLUTTextFont
  PEXLUTView

  Predefined entries are those automatically filled with valid data when a lookup table is created. The query is conducted based on the assumption that the lookup table would be used on drawables with the same root and depth as the specified drawable. The function returns at most the specified number of entries, starting with the specified entry index. The number of entries requested can not be larger than the number of predefined entries. Call PEXGetTableInfo(3) to determine the number of predefined entries.

  Entries in the returned list depend on the type specified. See PEXSetTableEntries(3) for the structure types.

  PEXlib allocates the memory for the returned table entries. PEXFreeTableEntries(3) should be called to deallocate the memory.
### DATA STRUCTURES

```c
#if NeedFunctionPrototypes
typedef void *PEXPointer;
#else
typedef char *PEXPointer;
#endif
```

### ERRORS

- **BadDrawable**
  - The specified drawable resource identifier is invalid.

- **BadMatch**
  - The specified drawable is unsupported.

- **BadValue**
  - The specified table type is invalid or unsupported, start is less than the minimum predefined entry, the sum of start and count is greater than the maximum predefined entry, or index 0 is not valid for the specified table type.

- **BadPEXLookupTable**
  - The specified table type is unsupported.

### SEE ALSO

- [PEXCreateLookupTable(3)]
- [PEXGetTableInfo(3)]
- [PEXGetDefinedIndices(3)]
- [PEXGetTableEntry(3)]
- [PEXGetTableEntries(3)]
- [PEXSetTableEntries(3)]
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXGetProtocolFloatFormat - Return Float Format used on Specified Display Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td>int PEXGetProtocolFloatFormat(Display *display)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>display</td>
</tr>
<tr>
<td>RETURNS</td>
<td>Protocol floating point format (see PEXGetEnumTypeInfo(3)); zero if unsuccessful.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>This function returns the protocol floating point format being used by PEXlib on the specified display connection.</td>
</tr>
<tr>
<td>ERRORS</td>
<td>None</td>
</tr>
</tbody>
</table>
**NAME**
PEXGetRendererAttributes - Get Renderer Attribute Values

**SYNTAX**
```c
PEXRendererAttributes *PEXGetRendererAttributes(Display *display, PEXRenderer renderer, unsigned long value_mask)
```

**PARAMETERS**
- `display` A pointer to a display structure returned by a successful `XOpenDisplay` call.
- `renderer` The resource identifier of the renderer to be queried.
- `value_mask` A mask indicating attributes to be returned from the renderer.

**RETURNS**
A pointer to the renderer attribute values; a null pointer if unsuccessful.

**DESCRIPTION**
This function returns attribute values from the specified `renderer` resource. The value mask indicates the attribute values to be returned. The value mask is constructed by OR’ing together the following constants:

- `PEXRABackgroundColor`
- `PEXRAInvisibilityIncl`
- `PEXRAClearImage`
- `PEXRALightTable`
- `PEXRAClearZ`
- `PEXRALineBundle`
- `PEXRAClipList`
- `PEXRAMarkerBundle`
- `PEXRAColorApproxTable`
- `PEXRAColorTable`
- `PEXRACurrentPath`
- `PEXRAPickExcl`
- `PEXRAPickIncl`
- `PEXRAEdgeBundle`
- `PEXRAPickStartPath`
- `PEXRAHLHSRMode`
- `PEXRAPipelineContext`
- `PEXRAHighlightExcl`
- `PEXRAHighlightIncl`
- `PEXRAInteriorBundle`
- `PEXRAViewTable`
- `PEXRAInvisibilityExcl`
- `PEXRAViewport`

PEXlib allocates the memory for the returned `renderer` attribute values. **PEXFreeRendererAttributes**(3) should be called to deallocate the memory.

**DATA STRUCTURES**
```c
typedef XID PEXRenderer;
```

See also PEXlib.h.

**ERRORS**
- `BadPEXRenderer`
  The specified `renderer` resource identifier is invalid.
- `BadValue`
  An invalid bit is set in the value mask.
SEE ALSO

PEXCreateRenderer(3)
PEXChangeRenderer(3)
PEXGetRendererDynamics(3)
PEXFreeRendererAttributes(3)
NAME
PEXGetRendererDynamics - Get Renderer Attribute Modification Dynamics

SYNTAX
Status PEXGetRendererDynamics(Display *display, PEXRenderer renderer, unsigned long *tables_return, unsigned long *name_sets_return, unsigned long *attributes_return)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
renderer The resource identifier of the renderer.
tables_return Returns a mask describing dynamics of lookup tables associated with the renderer.
name_sets_return Returns a mask describing dynamics of name sets associated with the renderer.
attributes_return Returns a mask describing dynamics of other attributes associated with the renderer.

RETURNS Zero if unsuccessful, non-zero otherwise.

DESCRIPTION
This function returns the modification dynamics for all of the attributes of the specified renderer resource. Each bit, in the returned bitmasks, indicates the dynamics for a particular renderer attribute. For each bit, a value of PEXDynamic indicates that the specified attribute may be modified at any time, and the change will take place immediately. A value of PEXNotDynamic indicates that the specified attribute may not be modified dynamically. In this case, the change is "pending" and will take effect at the next explicit or implicit PEXBeginRendering(3).

The pipeline context, pick start path, background color, clear image and clear z renderer attributes all take effect only at the time of "begin rendering or picking", and so do not have dynamics associated with them.

The bits in the tables bitmask are accessed with the following constants:
- PEXRDTColorApproxContents
- PEXRDTColorApproxTable
- PEXRDTColorTable
- PEXRDTColorTableContents
- PEXRDTDepthCueTable
- PEXRDTDepthCueTableContents
- PEXRDTEdgeBundle
- PEXRDTEdgeBundleContents
- PEXRDTInteriorBundle
- PEXRDTInteriorBundleContents
- PEXRDTLightTable
- PEXRDTLightTableContents
- PEXRDTLineBundle
The bits in the name sets bitmask are accessed with the following constants:

```c
PEXRDNHighlightNameSet
PEXRDNHighlightNameSetContents
PEXRDNInvisibilityNameSet
PEXRDNInvisibilityNameSetContents
PEXRDNPickNameSet
PEXRDNPickNameSetContents
```

The bits in the attributes bitmask are accessed with the following constants:

```c
PEXRDAClipList
PEXRDEchoMode
PEXRDAHLHSRMode
PEXRDANPCSubVolume
PEXRDAViewport
```

**ERRORS**

BadPEXRenderer

The specified `renderer` resource identifier is invalid.

**SEE ALSO**

PEXCreateRenderer(3)
PEXChangeRenderer(3)
PEXBeginRendering(3)
**NAME**
PEXGetSearchContext - Get Search Context Attributes

**SYNTAX**
PEXSCAttributes *PEXGetSearchContext(Display *display, PEXSearchContext context, unsigned long value_mask)

**PARAMETERS**
display A pointer to a display structure returned by a successful XOpenDisplay call.
context The resource identifier of the search context to be queried.
value_mask A mask indicating which attributes to return.

**RETURNS**
A pointer to the requested attribute values; a null pointer if unsuccessful.

**DESCRIPTION**
This function returns the requested attribute values of the specified search context resource. The value mask indicates which attributes are to be returned. The value mask is constructed by OR’ing together the following constants:
PEXSCCeiling
PEXSCDistance
PEXSCInvertedList
PEXSCModelClipFlag
PEXSCNormalList
PEXSCPosition
PEXSCStartPath

PEXlib allocates memory for the returned search context attribute values. PEXFreeSCAttributes(3) should be called to deallocate the memory.

**DATA STRUCTURES**
typedef XID PEXSearchContext;
typedef struct {
    PEXCoord position;
    float distance;
    unsigned short ceiling;
    Bool model_clip_flag;
    PEXStructurePath start_path;
    PEXListOfNameSetPair normal;
    PEXListOfNameSetPair inverted;
} PEXSCAttributes;
typedef struct {
    float x;
    float y;
    float z;
} PEXCoord;

modified November 1995 197
typedef struct {
    unsigned long count; /* number of elements */
    PEXElementRef *elements;
} PEXStructurePath;

typedef struct {
    PEXStructure structure;
    unsigned long offset;
} PEXElementRef;

typedef XID PEXStructure;

typedef struct {
    unsigned short count; /* number of pairs */
    PEXNameSetPair *pairs;
} PEXListOfNameSetPair;

typedef struct {
    PEXNameSet inclusion;
    PEXNameSet exclusion;
} PEXNameSetPair;

typedef XID PEXNameSet;

**ERRORS**

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BadPEXSearchContext</td>
<td>The specified search context resource identifier is invalid.</td>
</tr>
<tr>
<td>BadValue</td>
<td>An invalid bit is set in the value mask.</td>
</tr>
</tbody>
</table>

**SEE ALSO**

- PEXCreateSearchContext(3)
- PEXChangeSearchContext(3)
- PEXFreeSCAttributes(3)
NAME  PEXGetSizeOCs - Return the Protocol Formatted Size of Output Commands

SYNTAX  int PEXGetSizeOCs(int float_format, int oc_count, PEXOCData *oc_data)

PARAMETERS  float_format  The floating point format to use in computing the formatted size.
              oc_count   The number of output commands.
              oc_data    An array of output command data.

RETURNS  The size, in bytes, of the formatted output commands; zero if unsuccessful.

DESCRIPTION  This function returns information about the size of the protocol for the output commands. An unsuccessful return value is possible if the specified floating point format is invalid or unsupported by PEXlib.

DATA STRUCTURES  See PEXlib.h for the definition of PEXOCData.

ERRORS  None

modified November 1995
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXGetStructureInfo - Get Structure Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td>Status PEXGetStructureInfo(Display *display, PEXStructure structure, int float_format, unsigned long value_mask, PEXStructureInfo *info_return)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td><strong>display</strong></td>
</tr>
<tr>
<td></td>
<td><strong>structure</strong></td>
</tr>
<tr>
<td></td>
<td><strong>float_format</strong></td>
</tr>
<tr>
<td></td>
<td><strong>value_mask</strong></td>
</tr>
<tr>
<td></td>
<td><strong>info_return</strong></td>
</tr>
<tr>
<td>RETURNS</td>
<td>Zero if unsuccessful, non-zero otherwise.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>This function returns information about the specified structure resource. The value mask is constructed by OR’ing together the following constants:</td>
</tr>
<tr>
<td></td>
<td><strong>PEXEditMode</strong></td>
</tr>
<tr>
<td></td>
<td><strong>PEXElementPtr</strong></td>
</tr>
<tr>
<td></td>
<td><strong>PEXHasRefs</strong></td>
</tr>
<tr>
<td></td>
<td><strong>PEXLengthStructure</strong></td>
</tr>
<tr>
<td></td>
<td><strong>PEXNumElements</strong></td>
</tr>
<tr>
<td></td>
<td>The length of the structure is given in the number of 4-byte units, and is based upon the specified floating point format.</td>
</tr>
<tr>
<td>DATA STRUCTURES</td>
<td>typedef XID PEXStructure;</td>
</tr>
<tr>
<td></td>
<td>typedef struct {</td>
</tr>
<tr>
<td></td>
<td>unsigned long element_count;</td>
</tr>
<tr>
<td></td>
<td>unsigned long size;</td>
</tr>
<tr>
<td></td>
<td>Bool has_refs;</td>
</tr>
<tr>
<td></td>
<td>unsigned short edit_mode;</td>
</tr>
<tr>
<td></td>
<td>unsigned long element_pointer;</td>
</tr>
<tr>
<td></td>
<td>} PEXStructureInfo;</td>
</tr>
<tr>
<td>ERRORS</td>
<td><strong>BadPEXFloatingPointFormat</strong></td>
</tr>
<tr>
<td></td>
<td>The specified floating point format is invalid or unsupported.</td>
</tr>
<tr>
<td></td>
<td><strong>BadPEXStructure</strong></td>
</tr>
<tr>
<td></td>
<td>The specified structure resource identifier is invalid.</td>
</tr>
<tr>
<td></td>
<td><strong>BadValue</strong></td>
</tr>
<tr>
<td></td>
<td>An invalid bit is set in the value mask.</td>
</tr>
</tbody>
</table>
SEE ALSO PEXCreateStructure(3)
### NAME
PEXGetStructuresInNetwork - Get Structures in Network

### SYNTAX
```c
PEXStructure *PEXGetStructuresInNetwork(Display *display, PEXStructure structure, int which, unsigned long *count_return)
```

### PARAMETERS
- **display**: A pointer to a display structure returned by a successful `XOpenDisplay` call.
- **structure**: The resource identifier of the root structure in the structure network.
- **which**: A value indicating which structure resource identifiers to return (PEXAll or PEXOrphans).
- **count_return**: Returns the number of structure resource identifiers returned.

### RETURNS
An array of structure resource identifiers; a null pointer if unsuccessful.

### DESCRIPTION
This function returns a list of unique structure resource identifiers that are referenced in the structure network rooted at the specified `structure`. `PEXAll` indicates that identifiers of all structure resources referenced in the structure network are returned. `PEXOrphans` indicates that identifiers returned are those not referenced by any structures outside of those in the specified `structure` network.

The specified `structure` identifier will always be returned in the list, unless it is an invalid structure identifier, in which case the returned list will be empty.

PEXlib allocates memory for the return value. `XFree` should be called to deallocate the memory.

### DATA STRUCTURES
```c
typedef XID PEXStructure;
```

### ERRORS
- **BadPEXStructure**: The specified structure resource identifier is invalid.
- **BadValue**: The specified value for which parameter is invalid.

### SEE ALSO
- PEXCreateStructure(3)
- PEXGetAncestors(3)
- PEXGetDescendants(3)
NAME PEXGetTableEntries - Get Lookup Table Entries

SYNTAX Status PEXGetTableEntries(Display *display, PEXLookupTable table, unsigned int start,
unsigned int count, int value_type, int *table_type_return, PEXPointer
*entries_return)

PARAMETERS display A pointer to a display structure returned by a successful XOpenDisplay
call.
table The resource identifier of the lookup table from which the table entries
are to be obtained.
start The index of the first lookup table entry to be returned.
count The number of entries requested.
value_type The type of values to return (PEXSetValue or PEXRealizedValue).
table_type_return Returns the type of table.
entries_return Returns an array of table entries.

RETURNS Zero if unsuccessful, non-zero otherwise.

DESCRIPTION This function returns the values associated with a range of contiguous table indices, start-
ing at the specified entry index. If a table entry in the requested range is not defined, the
default entry for a table of this type is returned.

The type of entries in the returned list depend on the returned table type. See PEXSetTa-
bleEntries(3) for the structure types.

If the requested value type is PEXSetValue, the values returned will be those originally
set in the table entry. If the requested value type is PEXRealizedValue, the values
returned will be those actually used during rendering (i.e. the values used if the specified
entry value can not be exactly matched).

PEXlib allocates the memory for the returned entries. PEXFreeTableEntries(3) should be
called to deallocate the memory.

DATA STRUCTURES typedef XID PEXLookupTable;

#if NeedFunctionPrototypes
typedef void *PEXPointer;
#else
typedef char *PEXPointer;
#endif

ERRORS BadPEXLookupTable
The specified lookup table resource identifier is invalid, or the table type is
unsupported.
BadValue
The sum of start and count is too large, or index 0 is invalid for the specified table type.

SEE ALSO
PEXCreateLookupTable(3)
PGetXTableInfo(3)
PGetXPredefinedEntries(3)
PGetXDefinedIndices(3)
PGetXTableEntry(3)
PSetXTableEntries(3)
NAME  
PEXGetTableEntry - Get Lookup Table Entry

SYNTAX  
PEXPointer PEXGetTableEntry(Display *display, PEXLookupTable table, unsigned int index, int value_type, int *status_return, int *table_type_return)

PARAMETERS  
display  
A pointer to a display structure returned by a successful XOpenDisplay call.

table  
The resource identifier of the lookup table.

index  
The index of the entry to be returned from the lookup table.

value_type  
The type of values to return (PEXSetValue or PEXRealizedValue).

status_return  
Returns the entry status, either PEXDefinedEntry if the specified lookup table entry is defined or PEXDefaultEntry if it is undefined.

table_type_return  
Returns the type of table.

RETURNS  
A pointer to the lookup table entry; a null pointer if unsuccessful.

DESCRIPTION  
This function returns a single entry from the specified lookup table. If the specified lookup table entry is defined, the defined entry is returned. If the specified lookup table entry is not defined, the default entry for that type of table is returned.

The type of structure returned depends on the specified type. See PEXSetTableEntries(3) for the structure types.

If the requested value type is PEXSetValue, the values returned will be those originally set in the table entry. If the requested value type is PEXRealizedValue, the values returned will be those actually used during rendering (i.e. the values used if the specified entry value can not be exactly matched).

PEXlib allocates memory for the returned entry. PEXFreeTableEntries(3) should be called to deallocate the memory.

DATA STRUCTURES  
typedef XID PEXLookupTable;

#if NeedFunctionPrototypes
typedef void *PEXPointer;
#else
typedef char *PEXPointer;
#endif

ERRORS  
BadPEXLookupTable  
The specified lookup table resource identifier is invalid, or the table type is unsupported.

BadValue  
Index 0 is invalid for the specified table type.
<table>
<thead>
<tr>
<th>SEE ALSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEXCreateLookupTable(3)</td>
</tr>
<tr>
<td>PEXGetTableInfo(3)</td>
</tr>
<tr>
<td>PEXGetPredefinedEntries(3)</td>
</tr>
<tr>
<td>PEXGetDefinedIndices(3)</td>
</tr>
<tr>
<td>PEXGetTableEntries(3)</td>
</tr>
<tr>
<td>PEXSetTableEntries(3)</td>
</tr>
</tbody>
</table>
NAME

PEXGetTableInfo - Get Lookup Table Information

SYNTAX

Status PEXGetTableInfo(Display *display, Drawable drawable, int table_type, PEXTableInfo *info_return)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
drawable The resource identifier of a drawable.
table_type The type of lookup table (see the Description).
info_return Returns the lookup table information.

RETURNS

Zero if unsuccessful, non-zero otherwise.

DESCRIPTION

This function returns information about the specified type of lookup table. The type of lookup table must be one of the following:

PEXLUTColor
PEXLUTColorApprox
PEXLUTDepthCue
PEXLUTEdgeBundle
PEXLUTInteriorBundle
PEXLUTLight
PEXLUTLineBundle
PEXLUTMarkerBundle
PEXLUTPattern
PEXLUTTextBundle
PEXLUTTextFont
PEXLUTView

The returned information is based on the assumption that the lookup table would be used on drawables with the same root and depth as the specified drawable. The returned information includes the number of predefined table entries, the number of definable table entries, and the indices of the predefined minimum and maximum entries. Predefined entries are contiguous. The number of definable table entries includes the number of predefined entries since predefined table entries can be redefined by the application. All entries between the predefined minimum and maximum values are guaranteed to be defined initially (i.e. predefined entries have contiguous indices).

DATA STRUCTURES

typedef struct {
    unsigned short definable_entries;
    unsigned short predefined_count;
    unsigned short predefined_min;
    unsigned short predefined_max;
} PEXTableInfo;

modified November 1995
<table>
<thead>
<tr>
<th>ERRORS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BadDrawable</td>
<td>The specified drawable resource identifier is invalid.</td>
</tr>
<tr>
<td>BadMatch</td>
<td>The specified drawable is unsupported.</td>
</tr>
<tr>
<td>BadPEXLookupTable</td>
<td>The specified table type is unsupported.</td>
</tr>
<tr>
<td>BadValue</td>
<td>The specified table type is invalid.</td>
</tr>
</tbody>
</table>

**SEE ALSO**: PEXCreateLookupTable(3)
NAME  PEXGetWorkstationAttributes - Get Workstation Attribute Values

SYNTAX  PEXWorkstationAttributes ∗PEXGetWorkstationAttributes(Display ∗display, PEXWorkstation workstation, unsigned long ∗value_mask)

PARAMETERS  
  display  A pointer to a display structure returned by a successful XOpenDisplay call.
  workstation  The resource identifier of the workstation.
  value_mask  A mask indicating which workstation attribute values to return.

RETURNS  A pointer to the workstation attribute values; a null pointer if unsuccessful.

DESCRIPTION  This function returns the specified workstation's attributes. The value mask indicates which attributes are to be returned. PEXSetPWAttributeMask(3) or PEXSetPWAttributeMaskAll(3) should be used to setup the value mask.

PEXlib allocates memory for the returned workstation attribute values. PEXFreeWorkstationAttributes(3) should be called to deallocate the memory.

DATA STRUCTURES  typedef XID  PEXWorkstation;

typedef struct {
  short  drawable_update;
  int  visual_state;
  int  drawable_surface;
  int  view_update;
  PEXListOfView  defined_views;
  int  wks_update;
  PEXNPCSubVolume  req_npc_subvolume;
  PEXNPCSubVolume  cur_npc_subvolume;
  PEXViewport  req_workstation_viewport;
  PEXViewport  cur_workstation_viewport;
  int  hlhsr_update;
  PEXEnumTypeIndex  req_hlhsr_mode;
  PEXEnumTypeIndex  cur_hlhsr_mode;
  Drawable  drawable;
  PEXLookupTable  marker_bundle;
  PEXLookupTable  text_bundle;
  PEXLookupTable  line_bundle;
  PEXLookupTable  interior_bundle;
  PEXLookupTable  edge_bundle;
  PEXLookupTable  color_table;
  PEXLookupTable  depth_cue_table;
  PEXLookupTable  light_table;
  PEXLookupTable  color_approx_table;
} PEXWorkstation;
typedef struct {
    unsigned short count; /* number of views */
    PEXTableIndex *views;
} PEXListOfView;

typedef unsigned short PEXTableIndex;

typedef struct {
    PEXCoord min;
    PEXCoord max;
    XNPCSubVolume;
} PEXViewport;

typedef struct {
    float x;
    float y;
    float z;
} PEXCoord;

typedef struct {
    PEXDeviceCoord min;
    PEXDeviceCoord max;
    PEXSwitch use_drawable;
    unsigned char reserved[3];
} PEXViewport;

typedef struct {
    short x;
    short y;
    float z;
} PEXDeviceCoord;

typedef unsigned char PEXSwitch;
typedef short PEXEnumTypeIndex;
typedef XID PEXLookupTable;
typedef XID PEXNameSet;

typedef struct {
  unsigned long count; /* number of posted structures*/
  PEXPostedStructure *structures;
} PEXListOfPostedStructure;

typedef struct {
  PEXStructure sid;
  float priority;
} PEXPostedStructure;

typedef XID PEXStructure;

**ERRORS**

BadPEXWorkstation
The specified workstation resource identifier is invalid.

BadValue
An invalid bit is set in the value mask.

**SEE ALSO**

PEXGetWorkstationDynamics(3)
PEXSetPWAttributeMask(3)
PEXSetPWAttributeMaskAll(3)
NAME
PEXGetWorkstationDynamics - Get Workstation Attribute Modification Dynamics

SYNTAX
Status PEXGetWorkstationDynamics(Display *display, Drawable drawable, PEXWorksta-
tionDynamics *dynamics_return)

PARAMETERS
display  A pointer to a display structure returned by a successful XOpenDisplay call.
drawable The resource identifier of a drawable.
dynamics_return Returns the dynamics of the specified workstation.

RETURNS
Zero if unsuccessful, non-zero otherwise.

DESCRIPTION
This function returns the dynamics of the specified PHIGS workstation resource. Each
entry in the returned list has one of the following values: PEXIMM, PEXCBS or PEXIRG.
PEXIMM means that modification of the corresponding attribute will result in the correct
image displayed immediately without regeneration. PEXCBS means that modification of
the corresponding attribute will result in a simulation displayed immediately if the
display update mode is PEXSimulateSome. PEXIRG means that a regeneration is
required for the change to be visible.

DATA STRUCTURES
typedef struct {
    unsigned char view_rep;
    unsigned char marker_bundle;
    unsigned char text_bundle;
    unsigned char line_bundle;
    unsigned char interior_bundle;
    unsigned char edge_bundle;
    unsigned char color_table;
    unsigned char pattern_table;
    unsigned char wks_transform;
    unsigned char highlight_filter;
    unsigned char invisibility_filter;
    unsigned char hlhsr_mode;
    unsigned char structure_modify;
    unsigned char post_structure;
    unsigned char unpost_structure;
    unsigned char delete_structure;
    unsigned char reference_modify;
    unsigned char buffer_modify;
    unsigned char light_table;
    unsigned char depth_cue_table;
    unsigned char color_approx_table;
} PEXWorkstationDynamics;
<table>
<thead>
<tr>
<th>ERRORS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>BadDrawable</strong></td>
</tr>
<tr>
<td></td>
<td>The specified drawable resource identifier is invalid.</td>
</tr>
<tr>
<td></td>
<td><strong>BadMatch</strong></td>
</tr>
<tr>
<td></td>
<td>The specified drawable is unsupported.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SEE ALSO</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>PEXGetWorkstationAttributes(3)</strong></td>
</tr>
</tbody>
</table>
## NAME
PEXGetWorkstationPostings - Get Workstation Postings

## SYNTAX
```
Status PEXGetWorkstationPostings(Display *display, PEXStructure structure, unsigned long *count_return, PEXWorkstation **postings_return)
```

## PARAMETERS
- `display` A pointer to a display structure returned by a successful `XOpenDisplay` call.
- `structure` The resource identifier of the structure.
- `count_return` Returns the number of workstation identifiers.
- `postings_return` Returns an array of workstation resource identifiers.

## RETURNS
Zero if unsuccessful, non-zero otherwise.

## DESCRIPTION
This function returns a list of workstation resources to which the specified `structure` has been posted.

PEXlib allocates memory for the return value. `XFree` should be called to deallocate the memory.

## DATA STRUCTURES
```
typedef XID PEXStructure;
typedef XID PEXWorkstation;
```

## ERRORS
- `BadPEXStructure` The specified structure resource identifier is invalid.

## SEE ALSO
- `PEXPostStructure(3)`
- `PEXUnpostStructure(3)`
- `PEXUnpostAllStructures(3)`

---

modified November 1995
NAME
PEXGetWorkstationViewRep - Get Workstation View Representation

SYNTAX
Status PEXGetWorkstationViewRep(Display *display, PEXWorkstation workstation, unsigned int index, int *update_return, PEXViewRep *req_view_return, PEXViewRep *cur_view_return)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
workstation The resource identifier of the workstation.
index The view table index.
update_return Returns the update status of the view table index.
req_view_return Returns the requested value of the view table index.
cur_view_return Returns the current value of the view table index.

RETURNS
Zero if unsuccessful, non-zero otherwise.

DESCRIPTION
The function returns the update state, and the requested and current values for the specified view index of the specified workstation. The update will be PEXPending if a view change has been requested but not established; otherwise it will be PEXNotPending. If the specified view index is not defined, an error will be generated and the contents of the reply parameters will be undefined.

DATA STRUCTURES
typedef XID PEXWorkstation;

typedef struct {
    PEXTableIndex index;
    unsigned short reserved;
    PEXViewEntry view;
} PEXViewRep;

typedef unsigned short PEXTableIndex;

typedef struct {
    unsigned short clip_flags;
    unsigned short reserved;
    PEXNPCSubVolume clip_limits;
    PEXMatrix orientation;
    PEXMatrix mapping;
} PEXViewEntry;

typedef struct {
    PEXCoord min;
    PEXCoord max;
} PEXNPCSubVolume;

modified November 1995
typedef struct {
    float x;
    float y;
    float z;
} PEXCoord;

typedef float PEXMatrix[4][4];

**ERRORS**

**BadPEXWorkstation**
The specified *workstation* resource identifier is invalid.

**BadValue**
The specified view table entry is not defined.

**SEE ALSO**
PEXSetWorkstationViewRep(3)
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXIdentityMatrix - utility function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td>void PEXIdentityMatrix(PEXMatrix *transform_return)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td><em>transform_return</em></td>
</tr>
<tr>
<td></td>
<td>The returned identity matrix.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>This function returns an identity matrix.</td>
</tr>
<tr>
<td>ERRORS</td>
<td>None</td>
</tr>
<tr>
<td>NAME</td>
<td>PEXIdentityMatrix2D - utility function</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>SYNTAX</td>
<td>void PEXIdentityMatrix2D(PEXMatrix3x3 transform_return)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>transform_return</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>The returned identity matrix.</td>
</tr>
<tr>
<td>ERRORS</td>
<td>None</td>
</tr>
</tbody>
</table>
NAME  
PEXInitialize - Initialize PEXlib display connection

SYNTAX  
int PEXInitialize(Display *display, PEXExtensionInfo **info_return, int length, char *error_string)

PARAMETERS  
display  
A pointer to a display structure returned by a successful XOpenDisplay call.

info_return  
Returns a pointer to the extension information; if available (see the Description).

length  
The length, in bytes, of the memory allocated for the error string.

error_string  
A pointer to memory allocated for the error string.

RETURNS  
Zero if successful; otherwise, one of the following return values:

PEXBadExtension
PEXBadProtocolVersion
PEXBadFloatConversion
PEXBadLocalAlloc

DESCRIPTION  
PEXInitialize initializes PEXlib for the specified display.  
PEXInitialize can be called multiple times; subsequent calls will result in the same return  
value as the first call.

Standard PEXInitialize failure return values are:

PEXBadExtension - the PEX server extension does not exist,
PEXBadProtocolVersion - the PEX server extension does not support a compatible protocol version,
PEXBadFloatConversion - the PEX server extension does not support a protocol floating point format compatible with PEXlib's native format or a format to which PEXlib supports conversion, or
PEXBadLocalAlloc - PEXlib client-side allocation failed.

If PEXInitialize is successful (return value is zero), or if the return value is PEXBadProtocolVersion, a pointer to the extension information is returned in info_return. Otherwise, a NULL pointer is returned in info_return. The extension information is private to PEXlib and must not be modified or freed by the application.

The error string parameter specifies an address to memory allocated by the application.  
The constant PEXErrorStringLength is defined as a guideline for the size to allocate for  
the error string. If no failure occurred, the memory addressed by the error string parameter will be unchanged. If a failure does occur, an error string giving further information about the failure will be copied into this memory (up to the maximum specified by the length parameter).
The actual string returned is implementation dependent, and is provided primarily for convenience in printing an error message for the application’s end-user.

```c
typedef struct {
    unsigned short major_version;
    unsigned short minor_version;
    unsigned long release;
    unsigned long subset_info;
    char *vendor_name;
    int major_opcode;
    int first_event;
    int first_error;
} PEXExtensionInfo;
```

**DATA STRUCTURES**

**ERRORS** None

**SEE ALSO**

PEXGetExtensionInfo(3)
PEXGetEnumTypeInfo(3)
PEXGetImpDepConstants(3)
PEXMatchRenderingTargets(3)
NAME       PEXInvertMatrix - utility function
SYNTAX     int PEXInvertMatrix(PEXMatrix transform, PEXMatrix transform_return)
PARAMETERS
            transform     The transformation matrix to invert.
            transform_return The inverse transformation.
RETURNS     Zero if successful; non-zero if unsuccessful.
DESCRIPTION This function computes the inverse of a transformation matrix.
            An unsuccessful status is returned if the matrix is non-invertible.
            The two arguments may be the same variable, in which case the
            inversion is performed in-place, overwriting the original transform.
ERRORS      None
SEE ALSO    PEXInvertMatrix2D(3)
NAME  PEXInvertMatrix2D - utility function

SYNTAX  int PEXInvertMatrix2D(PEXMatrix3x3 transform, PEXMatrix3x3 transform_return)

PARAMETERS  

transform  The transformation matrix to invert.

transform_return  The inverse transformation.

RETURNS  Zero if successful; non-zero if unsuccessful.

DESCRIPTION  This function computes the inverse of a 2D transformation matrix. An unsuccessful status is returned if the matrix is non-invertible. The two arguments may be the same variable, in which case the inversion is performed in-place, overwriting the original transform.

ERRORS  None

SEE ALSO  PEXInvertMatrix(3)
**NAME**  
PEXLabel - Structure Label

**SYNTAX**  
void PEXLabel(Display ∗display, XID resource_id, PEXOCRequestType req_type, long label)

**PARAMETERS**
- **display**: A pointer to a display structure returned by a successful XOpenDisplay call.
- **resource_id**: The resource identifier of the renderer or structure.
- **req_type**: The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- **label**: A value to be used as a label.

**RETURNS**  
None

**DESCRIPTION**
This function creates a label output command which has no visible effect. Its primary function comes when it is stored in a structure, as labels can be used in various element pointer positioning operations. Labels within a particular structure need not be unique.

**ERRORS**
- **BadPEXRenderer**: The specified renderer resource identifier is invalid.
- **BadPEXStructure**: The specified structure resource identifier is invalid.

**SEE ALSO**
PEXDeleteToLabel(3)  
PEXDeleteBetweenLabels(3)  
PEXSetElementPtrAtLabel(3)
PEXListFonts - List PEX Fonts

char **PEXListFonts(Display *display, char *pattern, unsigned int max_names, unsigned long *count_return)

A pointer to a display structure returned by a successful XOpenDisplay call.

A null-terminated string pattern.

The maximum number of names to be returned.

Returns the actual number of font names.

An array of font names (null-terminated strings); a null pointer if unsuccessful or if no PEX fonts supported.

This function is like X11 XListFonts except only information about fonts that can support the full range of PEX text attributes is returned (i.e. only those fonts that are "PEX usable"). This list may or may not contain some of the same fonts returned by the X11 XListFonts function. This function returns a list of entries, each of which contains the name of a font matching the pattern. The pattern is a string that uses the ISO Latin-1 encoding and case is not significant in matching names. The '?' character (octal value 077) matches any single character, and the character '*' (octal value 052) matches any number of characters. The returned names are null-terminated, in lower-case and are also ISO Latin-1 encoded strings.

PEXlib allocates memory for the returned list of font names. PEXFreeFontNames should be called to deallocate the memory.

None

PEXListFontsWithInfo
PEXFreeFontNames

modified November 1995
NAME
PEXListFontsWithInfo - List PEX Fonts With Information

SYNTAX
char **PEXListFontsWithInfo(Display *display, char *pattern, unsigned int max_names,
unsigned long *count_return, PEXFontInfo **info_return)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
pattern A null-terminated string pattern.
max_names The maximum number of names to be returned.
count_return Returns the actual number of font names.
font_info Returns an array of font info structures (one for each name).

RETURNS
An array of font names (null-terminated strings); a null pointer if unsuccessful or if no PEX fonts supported.

DESCRIPTION
This function is like X11 XListFontsWithInfo except that only information about fonts that can support the full range of PEX text attributes is returned (i.e. those fonts that are "PEX usable"). This list may or may not contain some of the same fonts returned the X11 XListFontsWithInfo function. This function returns a list of entries, each of which contains information about a font matching the pattern. The pattern is a string that uses the ISO Latin-1 encoding and case is not significant in matching names. The '?' character (octal value 077) matches any single character, and the character '*' (octal value 052) matches any number of characters. The returned names are null-terminated, in lower case and are also ISO Latin-1 encoded strings.
The information returned for each font is identical to what PEXQueryFont(3) would return.

PEXlib allocates memory for the returned list of font names and font info. PEXFreeFontNames(3) should be called to deallocate the memory for the font names. PEXFreeFontInfo(3) should be called to deallocate the memory for the font information.

DATA STRUCTURES
typedef struct {
unsigned long first_glyph;
unsigned long last_glyph;
unsigned long default_glyph;
Bool all_exist;
Bool stroke;
unsigned short count; /* number of properties */
PEXFontProp *props;
} PEXFontInfo;

typedef struct {
Atom name;
unsigned long value;
}...
ERRORS
None

SEE ALSO
PEXListFonts(3)
PEXQueryFont(3)
PEXFreeFontNames(3)
PEXFreeFontInfo(3)
NAME
PEXLoadFont - Load PEX Font

SYNTAX
PEXFont PEXLoadFont(Display *display, char *font_name)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
font_name The font name (null-terminated string).

RETURNS
The resource identifier of the loaded PEX font.

DESCRIPTION
This function loads the specified PEX font, if necessary. The font name should use the ISO Latin-1 encoding and case is not significant in matching names. PEX fonts are not associated with a particular screen, and can be used with any renderer or PHIGS workstation resource. An error will be generated if the specified font is not "PEX usable", that is, it is not capable of supporting the full range of PEX text attributes.

DATA STRUCTURES
typedef XID PEXFont;

ERRORS
BadAlloc
The server failed to allocate the resource.
BadPEXFont
The specified font name does not name a usable PEX font.

SEE ALSO
PEXUnloadFont(3)

modified November 1995
### NAME
PEXLookAtViewMatrix - utility function

### SYNTAX
```c
int PEXLookAtViewMatrix(PEXCoord *from, PEXCoord *to, PEXVector *up, PEXMatrix matrix_return)
```

### PARAMETERS
- **from**: Viewing position, in world coordinates.
- **to**: Look at position, in world coordinates.
- **up**: Vector representing the up direction, in world coordinates.
- **matrix_return**: Matrix in which result is stored.

### RETURNS
Zero if successful; otherwise, one of the following:

- **PEXBadVectors**: The `from` and `to` arguments are equal, or the line between them is parallel with the up vector.
- **PEXBadVector**: `Up` is zero length.

### DESCRIPTION
This function creates a view orientation transform that defines the viewing direction and orientation. It is a slightly more intuitive interface to `PEXViewOrientationMatrix(3)`. The `from` position defines the viewpoint, and the `to` position specifies the point being viewed. These two parameters together define the view reference point (the VRC origin) and the view plane normal of `PEXViewOrientationMatrix(3)`. The view reference point is the `to` point; the view plane normal is the vector from `to` to `from`.

The view up vector is a 3D vector defined in world coordinates relative to the `to` point. The projection of this vector onto the plane through the `to` point and perpendicular to the view plane normal defines the Y axis of VRC.

### ERRORS
None

### SEE ALSO
- `PEXPolarViewMatrix(3)`
- `PEXViewOrientationMatrix(3)`
- `PEXViewMappingMatrix(3)`
NAME
PEXMapDCToWC - Map Device Coordinate Points to World Coordinate Points

SYNTAX
Status PEXMapDCToWC(Display ∗display, PEXWorkstation workstation, unsigned long
dc_count, PEXDeviceCoord ∗dc_points, unsigned int ∗view_index_return, unsigned
long ∗wc_count_return, PEXCoord ∗∗wc_points_return)

PARAMETERS
display
A pointer to a display structure returned by a successful XOpenDisplay
call.
workstation
The resource identifier of the workstation.
dc_count
The number of device coordinate points.
dc_points
An array of device coordinates.
view_index_return
Returns the view index of the view containing most or all of the points.
wq_count_return
Returns the number of world coordinate points.
wq_points_return
Returns an array of world coordinate points.

RETURNS
Zero if unsuccessful, non-zero otherwise.

DESCRIPTION
This function maps the device coordinate points to world coordinate points using the
specified workstation. Each view in the workstation’s current view table, which has an
inverse, is checked to see if it contains all the specified device coordinate points. (If the
view transform has no inverse, it is not considered.) The index of the view with the
highest view transformation input priority that contains all of the points is returned. If
no view contains all the points, the index of the view containing the most points is
returned. The points are transformed to world coordinates by passing them through the
inverse of the view transform associated with the view index. Points that are clipped
(outside the viewport) will not be transformed and returned, so the number of points
returned may be less than the number sent.

PEXlib allocates memory for the return value. XFree should be called to deallocate the
memory.

DATA STRUCTURES
typedef XID PEXWorkstation;
typedef struct {
    short x;
    short y;
    float z;
} PEXDeviceCoord;
typedef struct {
    float x;
    float y;
}
float z;
} PEXCoord;

**ERRORS**

**BadPEXWorkstation**
The specified *workstation* resource identifier is invalid.

**SEE ALSO**

PEXMapWCToDC(3)
PEXSetWorkstationViewPriority(3)
NAME
PEXMapWCToDC - Map World Coordinate Points to Device Coordinate Points

SYNTAX
Status PEXMapWCToDC(Display *display, PEXWorkstation workstation, unsigned long wc_count, PEXCoord *wc_points, unsigned int view_index, unsigned long *dc_count_return, PEXDeviceCoord **dc_points_return)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
workstation The resource identifier of the workstation.
wc_count The number of world coordinate points.
wc_points An array of world coordinate points.
view_index The view index to use in transforming the world coordinate points.
dc_count_return Returns the number of device coordinate points.
dc_points_return Returns an array of device coordinate points.

RETURNS
Zero if unsuccessful, non-zero otherwise.

DESCRIPTION
This function maps the world coordinate points to device coordinate points using the specified workstation and view index. The points are transformed to device coordinates by passing them through the view transform associated with the view index. Points that are clipped will not be returned, so the number of points returned may be less than the number sent.

PEXlib allocates memory for the return value. XFree should be called to deallocate the memory.

DATA STRUCTURES
typedef XID PEXWorkstation;

typedef struct {
    short x;
    short y;
    float z;
} PEXDeviceCoord;

typedef struct {
    float x;
    float y;
    float z;
} PEXCoord;

ERRORS
BadPEXWorkstation
The specified workstation resource identifier is invalid.

modified November 1995
SEE ALSO
PEXMapDCToWC(3)
PEXSetWorkstationViewPriority(3)
NAME
PEXMapXCToNPC - utility function

SYNTAX
int PEXMapXCToNPC(int point_count, PEXDeviceCoord2D *points, unsigned int
  window_height, double z_dc, PEXDeviceCoord *viewport, PEXNPCSubVolume
  *npc_sub_volume, int view_count, PEXViewEntry *views, int *view_return, int
  *count_return, PEXCoord *points_return)

PARAMETERS
point_count  The number of points to transform.
points       A pointer to an array of points to transform. The X and Y coordinates of
             these points are in drawable coordinates (XC). The Z coordinate is in
             device coordinates (DC).
window_height The height of the drawable.
z_dc         The z DC coordinate to assign the drawable points when converting
             them to DC.
viewport     An array of two device coordinate points defining a viewport, typically
             that of a renderer resource. The first point in the array is the lower-left
             corner of the viewport; the second point is the upper-right.
npc_sub_volume A pointer to an NPC subvolume, typically that of a renderer resource.
view_count   The number of views to search.
views        The view entries to search for inclusion of the transformed points.
view_return  Returns the view found to contain the most points.
count_return Returns the number of points contained in the returned view, or the
             number of points transformed if no views are specified.
points_return Returns a pointer to an array in which to store the transformed points.

RETURNS  Zero if successful; otherwise, one of the following:
          PEXBadViewport
          PEXBadSubVolume

DESCRIPTION  This function maps a list of drawable coordinates (XC) to NPC, and searches
              a specified list of view entries to determine the view containing the computed NPC points.

              The XC points are first transformed to DC, using the specified window height and assigning
              them the specified z DC value. They are then transformed to NPC by the viewport-
              to-subvolume transform implied by the specified viewport and NPC subvolume. The
              specified list of views is then searched, in order from 0 to the number of views minus 1,
              and the index of the first view containing all the NPC points is returned. If no view con-
              tains all the points, then the lowest-index view containing the most points is returned. In
              this case, only the points within the view are returned in points_return.

              When determining the containing view, only the clipping limits of the view are con-
              sidered, with no consideration given to the clipping flags or the viewing transforms.

modified November 1995
If no views are specified, the XC points are simply transformed to NPC points and returned. The value of the returned view is undefined in this case.

The viewport-to-subvolume transformation maps to NPC the largest region of the specified viewport that has the same aspect ratio as the NPC subvolume and is anchored at the back lower-left of the viewport (the corner of the viewport with the minimum X, Y and Z coordinates). Points that lie outside this region of the viewport are not transformed.

When specifying NPC and DC, the X, Y and Z limits must be as follows:

\[
\text{xmin < xmax, ymin < ymax, zmin} \leq \text{zmax}
\]

**ERRORS**  
None

**SEE ALSO**  
PEXMapXCToNPC2D(3)  
PEXNPCToXCTransform(3)  
PEXXCToNPCTransform(3)
NAME
PEXMapXCToNPC2D - utility function

SYNTAX
int PEXMapXCToNPC2D(int point_count, PEXDeviceCoord2D *points, unsigned int window_height, PEXDeviceCoord2D *viewport, PEXNPCSubVolume *npc_sub_volume, int view_count, PEXViewEntry *views, int *view_return, int *count_return, PEXCoord2D *points_return)

PARAMETERS
point_count The number of points to transform.
points A pointer to an array of drawable-coordinate (XC) points to transform.
window_height The height of the drawable.
viewport An array of two device coordinate points defining a viewport, typically that of a renderer resource. The first point in the array is the lower-left corner of the viewport; the second point is the upper-right.
npc_sub_volume A pointer to an NPC subvolume, typically that of a renderer resource.
view_count The number of views to search.
views The view entries to search for inclusion of the transformed points.
view_return The view found to contain the most points.
count_return Returns the number of points contained in the returned view, or the number of points transformed if no views are specified.
points_return A pointer to an array in which to store the transformed points.

RETURNS
Zero if successful; otherwise, one of the following:
PEXBadViewport
PEXBadSubVolume

DESCRIPTION
This function maps a list of drawable coordinates (XC) to NPC, and searches a specified list of view entries to determine the view containing the computed NPC points.

The XC points are first transformed to 2D DC, using the specified window height, then transformed to 2D NPC by the viewport-to-subvolume transform implied by the specified viewport and NPC subvolume. The specified list of views is then searched, in order from 0 to the number of views minus 1, and the index of the first view containing all the NPC points is returned. If no view contains all the points, then the lowest-index view containing the most points is returned. In this case, only the points within the view are returned.

When determining the containing view, only the x-y clipping limits of the view are considered, with no consideration given to the front and back clipping limits, the clipping flags, or the viewing transforms.

If no views are specified, the XC points are simply transformed to NPC points and returned. The value of the returned view is undefined in this case.

modified November 1995
The viewport-to-subvolume transformation maps to NPC the largest region of the specified viewport that has the same aspect ratio as the NPC subvolume and is anchored at the back lower-left of the viewport (the corner of the viewport with the minimum X, Y and Z coordinates). Points that lie outside this region of the viewport are not transformed.

When specifying NPC and DC, the X, Y and Z limits must be as follows:

- xmin < xmax, ymin < ymax, zmin <= zmax

**ERRORS**
None

**SEE ALSO**
- PEXMapXCToNPC(3)
- PEXNPCToXCTransform2D(3)
- PEXXCToNPCTransform2D(3)
NAME
PEXMarkers - 3D Markers Primitive

SYNTAX
void PEXMarkers(Display *display, XID resource_id, PEXOCRequestType req_type,
unsigned int count, PEXCoord *points)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay
call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOC-
Store, PEXOCRenderSingle or PEXOCStoreSingle).
count The number of points.
points An array of points specifying marker locations.

RETURNS
None

DESCRIPTION
This function creates a markers output primitive.
A marker is a geometric primitive with a position as its only geometric attribute. The
position is specified in modeling coordinates.
Rendering transforms the marker’s position to a position in device coordinates. A
marker has no geometric size, so geometric transformations do not affect the displayed
size of the marker. The marker’s color is affected only by depth-cueing and mapped to a
device color. The clipping of markers whose position is inside the clipping volume but
whose rendering is outside the clipping volume is implementation-dependent.
Depending on the setting of the marker ASF attributes, the marker color, marker type,
and marker scale attributes are obtained either directly from the current marker attrib-
utes or from the marker bundle.

DATA STRUCTURES
See PEXlib.h.

ERRORS
BadPEXRender
The specified renderer resource identifier is invalid.
BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO
PEXSetMarkerType(3)
PEXSetMarkerScale(3)
PEXSetMarkerColorIndex(3)
PEXSetMarkerColor(3)
PEXSetMarkerBundleIndex(3)
NAME
PEXMarkers2D - 2D Markers Primitive

SYNTAX
void PEXMarkers2D(Display *display, XID resource_id, PEXOCRequestType req_type,
unsigned int count, PEXCoord2D *points)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
count The number of points.
points An array of points specifying marker locations.

RETURNS
None

DESCRIPTION
This function creates a 2D markers output primitive. This function is like PEXMarkers(3), except that the vertices consist of only x- and y-components. The z-component is assumed to be zero. This primitive is two-dimensional only in that the z-components are implied. Geometry transformations are still carried out in three dimensions.

DATA STRUCTURES
See PEXlib.h.

ERRORS
BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO
PEXSetMarkerType(3)
PEXSetMarkerScale(3)
PEXSetMarkerColorIndex(3)
PEXSetMarkerColor(3)
PEXSetMarkerBundleIndex(3)

modified November 1995
NAME
PEXMatchRenderingTargets - Return Information about Supported Rendering Targets

SYNTAX
Status PEXMatchRenderingTargets(Display *display, Drawable drawable, int depth, int
type, Visual *visual, unsigned long max_targets, unsigned long *count_return, PEX-
RenderingTarget **targets_return)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay
call.
drawable The drawable indicates which screen the application is interested in.
depth The depth of interest.
type The drawable type of interest (PEXAnyDrawable, PEXWindowDrawable,
PEXPixmapDrawable, PEXBufferDrawable).
visual The visual of interest.
max_targets The maximum number of targets to return.
count_return Returns the actual number of targets in the return array.
targets_return Returns an array of rendering target information.

RETURNS Zero if unsuccessful, non-zero otherwise.

DESCRIPTION This function returns information about which drawable types the server supports.
A drawable is specified only to indicate the screen for which the returned information
should apply. None of the other drawable attributes are used.
The depth value is specified to indicate the depth for which the returned information
should apply. If the application wants information on all supported depths, a value of
zero should be specified.
The drawable type is specified to indicate the type of drawable for which the returned
information should apply. The type field of the PEXRenderingTarget data structure will
have these same values with the exception of PEXAnyDrawable.
The visual is specified to indicate the visual for which the returned information should
apply. If the application wants information on all supported visuals, a null pointer
should be specified.
PEXlib allocates memory for the returned target values. XFree should be called to deallo-
cate the memory.

DATA STRUCTURES
typedef struct {
    int depth;
    int type; /* PEXWindowDrawable, PEXPixmapDrawable, PEXBufferDrawable */
    Visual *visual;
} PEXRenderingTarget;

modified November 1995
ERRORS

BadDrawable
The specified drawable resource identifier is invalid.

BadValue
The specified visual is invalid.
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXMatrixMult - utility function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td>void PEXMatrixMult(PEXMatrix matrix1, PEXMatrix matrix2, PEXMatrix matrix_return)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>matrix1 First matrix to be multiplied.</td>
</tr>
<tr>
<td></td>
<td>matrix2 Second matrix to be multiplied.</td>
</tr>
<tr>
<td></td>
<td>matrix_return Matrix into which result is stored.</td>
</tr>
<tr>
<td>RETURNS</td>
<td>None</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>Performs a matrix multiplication: matrix = matrix1 X matrix2</td>
</tr>
<tr>
<td></td>
<td>If the return matrix is the same as one of the input matrices, the function will overwrite the input values with the multiplied values.</td>
</tr>
<tr>
<td>ERRORS</td>
<td>None</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>PEXMatrixMult2D(3)</td>
</tr>
</tbody>
</table>
### NAME
PEXMatrixMult2D - utility function

### SYNTAX
```c
void PEXMatrixMult2D(PEXMatrix3x3 matrix1, PEXMatrix3x3 matrix2, PEXMatrix3x3 matrix_return);
```

### PARAMETERS
- `matrix1`: First matrix to be multiplied.
- `matrix2`: Second matrix to be multiplied.
- `matrix_return`: Matrix into which result is stored.

### RETURNS
None

### DESCRIPTION
Performs a 3x3 matrix multiplication: \( \text{matrix} = \text{matrix1} \times \text{matrix2} \).
If the return matrix is the same as one of the input matrices, the function will overwrite the input values with the multiplied values.

### ERRORS
None

### SEE ALSO
PEXMatrixMult(3)

modified November 1995
NAME
PEXNPCToXCTransform - utility function

SYNTAX
int PEXNPCToXCTransform(PEXNPCSubVolume *npc_sub_volume,
                         PEXDeviceCoord *viewport,
                         unsigned int window_height,
                         PEXMatrix transform_return)

PARAMETERS
npc_sub_volume A pointer to an NPC subvolume, typically that of a renderer resource.
viewport An array of two device coordinate points defining a viewport, typically
          that of a renderer resource. The first point in the array is the lower-left
          corner of the viewport; the second point is the upper-right.
window_height The height of the drawable.
transform_return The returned transformation.

RETURNS
Zero if successful; otherwise, one of the following:
    PEXBadViewport - (xmin >= xmax, or
    PEXBadSubVolume - (xmin >= xmax, or

DESCRIPTION
This function computes the transformation matrix to map an NPC point to a drawable
coordinate (XC), using the specified NPC subvolume, DC viewport, and drawable height.
The returned transformation matrix first applies the subvolume-to-viewport transforma-
tion, then transforms the x and y coordinates of the resulting points to drawable coordi-
nates, leaving the z coordinate in DC.

When specifying NPC and DC, the X, Y and Z limits must be as follows:
    xmin < xmax , ymin < ymax , zmin <= zmax

ERRORS
None

SEE ALSO
PEXNPCToXCTransform2D(3)

modified November 1995
NAME
PEXNPToXCTransform2D - utility function

SYNTAX
int PEXNPToXCTransform2D(PEXNPCSubVolume *npc_sub_volume,
PEXDeviceCoord2D *viewport, unsigned int window_height, PEXMatrix3x3
transform_return)

PARAMETERS
npc_sub_volume A pointer to an NPC subvolume, typically that of a renderer resource.
viewport An array of two device coordinate points defining a viewport, typically
that of a renderer resource. The first point in the array is the lower-left
corner of the viewport; the second point is the upper-right.
window_height The height of the drawable.
transform_return The returned transformation.

RETURNS
Zero if successful; otherwise, one of the following:
PEXBadViewport - (xmin >= xmax, or
PEXBadSubVolume - (xmin >= xmax, or

DESCRIPTION
This function computes the 2D transformation matrix to map a 2D NPC point to a 2D
drawable coordinate (XC), using the specified NPC subvolume, DC viewport, and draw-
able height. The returned transformation matrix first applies the subvolume-to-viewport
transformation, then transform the resulting points to drawable coordinates.
When specifying NPC and DC, the X, Y and Z limits must be as follows:
xmin < xmax , ymin < ymax , zmin <= zmax

ERRORS
None

SEE ALSO
PEXNPToXCTransform(3)
NAME
PEXNURBCurve - Non-Uniform Rational B-spline Curve Primitive

SYNTAX
void PEXNURBCurve(Display *display, XID resource_id, PEXOCRequestType req_type,
                    int rationality, int order, float *knots, unsigned int count,
                    PEXArrayOfCoord points, double tmin, double tmax)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
rationality The type of B-spline curve, whether rational or non-rational (PEXRational or PEXNonRational).
order The order of the polynomial expression.
knots An array of floats specifying the B-spline curve knots.
count The number of control points that define the curve.
points An array of control points defining the B-spline curve.
tmin The minimum parameter value at which to evaluate the curve.
tmax The maximum parameter value at which to evaluate the curve.

RETURNS
None

DESCRIPTION
This function creates a non-uniform B-spline curve output primitive.

The curve order is specified as a positive number. PEXGetImpDepConstants(3) can be called to determine the largest supported value for curve order. The spline shape is specified using a list of knots in the parametric coordinate space and a list of control points in modeling coordinates. The number of control points must be at least as large as the order. The number of knots is the sum of the curve order plus the number of control points. The sequence of knots must be non-decreasing,

i.e. t(0) <= t(1) <= ... <= t(k-1)

where k is the number of knots.

If the rationality is PEXRational, the control points must be specified in homogeneous (4D) modeling coordinates. PEXlib assumes that 4D modeling coordinates (x,y,z) have already been multiplied by the homogeneous coordinate (w). If the rationality is PEXNonRational, the control points must be specified in non-homogeneous (3D) modeling coordinates.

Evaluation of the spline is restricted to a specific region in the parametric coordinate space. The parametric limits, tmin and tmax, specify the region in the parametric coordinate space over which the spline is to be evaluated (tmin must be less than tmax). The parametric bounds must also satisfy the restriction tmin >= t(order), tmax <= t(k+1-order).
Depending on the ASF attributes, line color, line type, line width and curve approximation attributes are obtained either directly from the current line attributes or from the line bundle.

If the specified curve order is not supported, the output primitive is stored in a structure, but when rendered, the primitive is ignored and has no visual effect. Also, the curve order must not be less than one.

**DATA STRUCTURES**

typedef union {
    PEXCoord2D *point_2d;
    PEXCoord *point;
    PEXCoord4D *point_4d;
} PEXArrayOfCoord; /* Pointer to array of points */

See also PEXlib.h.

**ERRORS**

BadPEXOutputCommand
   The output command contains an invalid value.

BadPEXRenderer
   The specified renderer resource identifier is invalid.

BadPEXStructure
   The specified structure resource identifier is invalid.

**SEE ALSO**

PEXSetLineType(3)
PEXSetLineWidth(3)
PEXSetLineColorIndex(3)
PEXSetLineColor(3)
PEXSetLineBundleIndex(3)
PEXSetCurveApprox(3)
PEXGetImpDepConstants(3)
NAME | PEXNURBSurface - Non-Uniform Rational B-spline Surface Primitive

SYNTAX | void PEXNURBSurface(Display *display, XID resource_id, PEXOCRequestType req_type, int rationality, int uorder, int vorder, float *uknots, float *vknots, unsigned int col_count, unsigned int row_count, PEXArrayOfCoord points, unsigned int curve_count, PEXListOfTrimCurve *trim_curves)

PARAMETERS | display | A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id | The resource identifier of the renderer or structure.
req_type | The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
rationality | The type of B-spline surface, whether rational or non-rational (PEXRational or PEXNonRational).
uorder | The order of the polynomial expression in the u direction.
vorder | The order of the polynomial expression in the v direction.
uknots | An array of floats specifying the B-spline curve knots in the u direction.
vknots | An array of floats specifying the B-spline curve knots in the v direction.
col_count | The number of columns in the points array (number of points in the u direction).
row_count | The number of rows in the points array (number of points in the v direction).
points | An array of points defining the B-spline surface.
curve_count | The number of trimming curves.
trim_curves | A pointer to a list of trimming curves.

RETURNS | None

DESCRIPTION | This function creates a non-uniform B-spline surface output primitive. The surface is generated as a function of the parametric variables u and v. The u and v order must be positive integers and indicate the order of the surface in each of the u and v parameter dimensions. PEXGetImpDepConstants(3) can be called to determine the largest supported value for surface u and v order. The spline shape is specified using two lists of knots in the parametric coordinate space, plus an array of control points specified in modeling coordinates. The u and v knot sequences must each form a non-decreasing sequence of numbers. The column count indicates the number of control points in the u direction and the row count indicates the number of control points in the v direction. The control points are stored in the array in row-major order (i.e., the column number varies fastest as vertices are stored in the array) and the rows increase in the direction of increasing v. The number of knots in the u direction is the sum of the order in the u direction and column count. The number of knots in the v direction is the sum of...
the order in the v direction and row count. The number of control points in each direc-
tion must be at least as large as the corresponding order.

The minimum and maximum knot values define the range over which the B-spline sur-
face is evaluated in the u or v parametric direction.

If the rationality is **PEXRational**, the control point list must be specified in homogeneous
(4D) modeling coordinates. If rationality is **PEXNonRational**, the control point list must
be specified in non-homogeneous (3D) modeling coordinates.

In addition to the parametric bounds, a list of trimming loops may also be specified.
Trimming loops serve to further restrict the region in parametric coordinate space over
which the surface is evaluated. Each trimming loop is defined as a list of one or more B-
spline trimming curves that are connected head-to-tail. Each trim curve is a completely
specified NURB curve, i.e. it is rational or non-rational, has its own order, etc. The list
must be explicitly closed so that the tail of the last curve joins the head of the first. Each
trimming curve is parameterized independently. If there is floating point inaccuracy in
closure or in head-to-tail connectivity between curves, closure or connectivity will be
assumed. Trimming loops are defined in the parameter space of the surface and may not
go outside the parameter space of the surface.

When no trimming loops are specified, the rectangular parameter limits of the surface are
rendered as the edges of the surface based on the edge flag attribute.

Trimming loops define the region of the surface that is to be rendered based on the fol-
lowering two rules: (1) a point is in the portion of the surface to be rendered if any ray pro-
jected from it to infinity has an odd number of intersections with trimming loops, and (2)
traveling in the direction of a trimming loop, the portion of the surface to be trimmed
away should be on the right and the portion to be retained should be on the left. In other
words, a loop defined in counter-clockwise order will cause the interior of the loop to be
retained and the exterior to be trimmed away. A clockwise loop will cause the exterior of
the loop to be retained and the interior to be trimmed away. If loops are nested, they
must alternate in direction. In all cases, the outermost loop must be counter-clockwise.
No trimming curve may intersect itself and no trimming loop may intersect itself or any
other trimming loop. Trimming loops that do not obey these rules will result in
implementation-dependent behavior.

Each trimming curve has a visibility flag that controls its visibility for the purposes of
surface edge display. Depending on the surface edge attributes and the visibility flags
associated with trimming curves, the curves in trimming loops may be drawn as surface
edges.

All attributes affecting the representation of fill area sets also affect the representation of
the non-uniform B-spline surface primitive. In addition, the surface approximation is
used to determine how to approximate the B-spline surface and the parametric surface
characteristics are used to specify the appearance of the surface.

If either of the specified surface orders are not supported, the output primitive is stored
in a structure, but when rendered, the primitive is ignored and has no visual effect.
Trimming curve specification must abide by the constraints of NURB curve (e.g. number of control points at least as large as the order, non-decreasing knot sequence, order plus number of controls points equals the number of knots). Also, the trim curve order must not be less than two.

**DATA STRUCTURES**

```c
typedef union {
    PEXCoord2D *point_2d;
    PEXCoord *point;
    PEXCoord4D *point_4d;
} PEXArrayOfCoord; /* Pointer to array of points */
```

```c
typedef struct {
    unsigned short count; /* number of curves */
    PEXTrimCurve *curves;
} PEXListOfTrimCurve;
```

```c
typedef struct {
    PEXSwitch visibility; /* True or False */
    unsigned char reserved;
    unsigned short order;
    PEXCoordType rationality; /* PEXRational or PEXNonRational */
    PEXEnumTypeIndex approx_method; /* see PEXGetEnumTypeInfo */
    float tolerance;
    float tmin, tmax;
    PEXListOfFloat knots;
    unsigned short count; /* number of control points */
    PEXArrayOfCoord control_points;
} PEXTrimCurve;
```

```c
typedef unsigned char PEXSwitch;
typedef unsigned short PEXCoordType;
typedef short PEXEnumTypeIndex;
```

```c
typedef struct {
    unsigned short count; /* number of floats */
    float *floats;
} PEXListOfFloat;
```

See also **PEXlib.h.**

**ERRORS**

- **BadPEXOutputCommand**
  - The output command contains an invalid value.

- **BadPEXRenderer**
  - The specified renderer resource identifier is invalid.

**BadPEXStructure**

*modified November 1995 249*
The specified structure resource identifier is invalid.

SEE ALSO

PEXSetInteriorStyle(3)
PEXSetInteriorStyleIndex(3)
PEXSetSurfaceColorIndex(3)
PEXSetSurfaceColor(3)
PEXSetReflectionAttributes(3)
PEXSetReflectionModel(3)
PEXSetSurfaceInterpMethod(3)
PEXSetBFInteriorStyle(3)
PEXSetBFInteriorStyleIndex(3)
PEXSetBFSurfaceColorIndex(3)
PEXSetBFSurfaceColor(3)
PEXSetBFReflectionAttributes(3)
PEXSetBFReflectionModel(3)
PEXSetBFSurfaceInterpMethod(3)
PEXSetSurfaceApprox(3)
PEXSetFacetCullingMode(3)
PEXSetFacetDistinguishFlag(3)
PEXSetPatternSize(3)
PEXSetPatternAttributes(3)
PEXSetPatternAttributes2D(3)
PEXSetInteriorBundleIndex(3)
PEXSetSurfaceEdgeFlag(3)
PEXSetSurfaceEdgeType(3)
PEXSetSurfaceEdgeWidth(3)
PEXSetSurfaceEdgeColor(3)
PEXSetSurfaceEdgeColorIndex(3)
PEXSetEdgeBundleIndex(3)
PEXSetParaSurfCharacteristics(3)
PEXGetImpDepConstants(3)
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXNoop - Noop Output Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td>void PEXNoop(Display *display, XID resource_id, PEXOCRequestType req_type)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>display A pointer to a display structure returned by a successful XOpenDisplay call.</td>
</tr>
<tr>
<td></td>
<td>resource_id The resource identifier of the renderer or structure.</td>
</tr>
<tr>
<td></td>
<td>req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).</td>
</tr>
<tr>
<td>RETURNS</td>
<td>None</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>This function creates a no-op output command which, when rendered, will do nothing except update the current path.</td>
</tr>
<tr>
<td>ERRORS</td>
<td>BadPEXRenderer The specified renderer resource identifier is invalid.</td>
</tr>
<tr>
<td></td>
<td>BadPEXStructure The specified structure resource identifier is invalid.</td>
</tr>
</tbody>
</table>

modified November 1995
NAME  PEXNormalizeVectors - utility function

SYNTAX  int PEXNormalizeVectors(int count, PEXVector *vectors, PEXVector *vectors_return)

PARAMETERS  count  The number of vectors to normalize.
  vectors  A pointer to the array of vectors to normalize.
  vectors_return  A pointer to an array in which to store the normalized vectors.

RETURNS  Zero if successful; otherwise, one of the following:
  PEXBadVector  One of the vectors has zero magnitude.

DESCRIPTION  This function normalizes each vector in the specified list of 3D vectors. An error is returned if any vector in the list has a magnitude of zero. All non-zero vectors in the list are still normalized, however.
  If the return array is the same as the input array, the function will overwrite the input values with the normalized values.

ERRORS  None

SEE ALSO  PEXTransformVectors(3)  PEXNormalizeVectors2D(3)
**NAME**  
PEXNormalizeVectors2D - utility function

**SYNTAX**  
```c
int PEXNormalizeVectors2D(int count, PEXVector2D *vectors, PEXVector2D *vectors_return)
```

**PARAMETERS**  
- `count`  
  The number of vectors to normalize.
- `vectors`  
  A pointer to the array of vectors to normalize.
- `vectors_return`  
  A pointer to an array in which to store the normalized vectors.

**RETURNS**  
Zero if successful; otherwise, one of the following:
- `PEXBadVector`  
  One of the vectors has zero magnitude.

**DESCRIPTION**  
This function normalizes each vector in the specified list of 2D vectors. An error is returned if any vector in the list has a magnitude of zero. All non-zero vectors in the list are still normalized, however.

If the return array is the same as the input array, the function will overwrite the input values with the normalized values.

**ERRORS**  
None

**SEE ALSO**  
PEXTransformVectors2D(3)  
PEXNormalizeVectors(3)
## NAME
PEXOrthoProjMatrix - utility function

## SYNTAX
```c
int PEXOrthoProjMatrix(double height, double aspect, double near, double far, PEXMatrix matrix_return)
```

## PARAMETERS
- **height**: The height of the orthographic viewing box.
- **aspect**: The aspect ratio (width/height) of the orthographic viewing box.
- **near**: The distance, in view reference coordinates, from the VRC origin to the front clipping plane.
- **far**: The distance, in view reference coordinates, from the VRC origin to the back clipping plane.
- **matrix_return**: Matrix in which result is stored.

## RETURNS
Zero if successful; otherwise, one of the following:
- **PEXBadLimits**: The viewing box depth, width, or height is zero.

## DESCRIPTION
This routine formats a view mapping matrix.

A projection matrix defines a visible region of the coordinate space. An orthographic projection defines the visible region as a box specified by its height, width (the height multiplied by the aspect), and its near and far boundaries.

The reference point for the projection is the origin of VRC; the near and far clipping planes are defined with respect to it. The height is defined in view reference coordinates. Clipping at the planes is controlled by the clipping flags in the selected view table entry.

## ERRORS
None

## SEE ALSO
- PEXLookAtViewMatrix(3)
- PEXViewOrientationMatrix(3)
- PEXViewMappingMatrix(3)
- PEXPerspProjMatrix(3)
NAME
PEXPerspProjMatrix - utility function

SYNTAX
int PEXPerspProjMatrix(double fovy, double distance, double aspect, double near, double far, PEXMatrix matrix_return)

PARAMETERS
fovy  
Field of view (in radians) in the horizontal direction.

distance 
The distance to the eye-point.

aspect 
The aspect ratio (width/height) of the perspective viewing frustum.

near 
The distance to the near clipping plane.

far 
The distance to the far clipping plane.

matrix_return 
Matrix in which result is stored.

RETURNS
Zero if successful; otherwise, one of the following:

PEXBadLimits
near <= far, fovy = 0, aspect = 0, or distance <= near

DESCRIPTION
This routine formats a view mapping matrix.

A projection matrix defines the visible region of the coordinate space. A perspective projection defines the visible region as a truncated pyramid or frustum. The amount of perspective in the projection is specified by the field of view argument, fovy. The perspective increases as the angle increases to a value of pi radians.

The distance between the eye-point and the origin is specified by the distance. If the application program calls PEXLookAtViewMatrix(3) to calculate the view orientation matrix, the distance is typically the distance between the from and to points specified to that routine. If the application program calls PEXPolarViewMatrix(3) to calculate the view orientation matrix, the distance is typically the same distance specified to that routine.

The height of the frustum at the near clipping plane is determined by fovy and the distance to the near plane. The width of the frustum is determined from the aspect ratio.

The reference point for the projection is the origin of VRC; the near and far clipping planes are defined with respect to it. Clipping at the planes is controlled by the clip flags in the selected view table entry.

It is useful to think of PEXPerspProjMatrix as defining a camera. The object being viewed is defined near the origin. The lens is defined by fovy; a larger value of fovy defines a wide angle lens. For those who wish to keep the height at the near plane constant and automatically back up the camera to frame the subject, the relationship between the field of view, the eye distance, which is the distance between the eye-point and the near plane, and the height, which is the height at the near plane, is:

\[ \tan \left( \frac{fovy}{2} \right) = \left( \frac{\text{height}/2}{\text{eye_distance}} \right) \]

For example, if a unit cube is being viewed, a "look at" view with the to point at the center of the cube or a "polar" view with the viewed point at the center of the cube, places the cube at the origin. A matrix created by PEXPerspProjMatrix with aspect = 1, near = 0.5,
and far = -0.5 makes the entire cube visible, with the field of view and distance controlling the amount of perspective applied.

**ERRORS**  None

**SEE ALSO**  PEXLookAtViewMatrix(3)
PEXViewOrientationMatrix(3)
PEXViewMappingMatrix(3)
PEXOrthoProjMatrix(3)
NAME

PEXPickAll - Pick All Traversal

SYNTAX

PEXPickPath *PEXPickAll(Display *display, Drawable drawable, PEXRenderer renderer,
        int method, int max_hits, int pick_device_type, PEXPickRecord *pick_record, int *
        status_return, int *more_return, unsigned long *count_return)

PARAMETERS

display A pointer to a display structure returned by a successful XOpenDisplay call.
drawable The resource identifier of a drawable.
renderer The resource identifier of the renderer.
method The pick all method (PEXPickAllAll or PEXPickAllVisible).
max_hits The maximum number of hits to be returned.
pick_device_type The pick device type (PEXPickDeviceDCHitBox or PEXPickDeviceNP-
        CHitVolume).
pick_record A pointer to the pick data record.
status_return Returns the status of the pick operation.
more_return Returns the status of remaining picks.
count_return Returns the number of pick paths.

RETURNS

An array of pick paths; a null pointer if unsuccessful or no pick (see also status_return).

DESCRIPTION

This function traverses the structure network specified by the renderer’s current pick start path. Hit testing begins after the last element specified in the renderer’s current pick start path and continues until one of two conditions is met: the maximum number of hits is reached, or the last element of the first structure in the pick start path is processed. If the pick start path does not define a valid hierarchical path, a BadPEXPath error is sent and a null pick path is returned.

Standard pick all methods are PEXPickAllAll and PEXPickAllVisible. The supported pick device types are inquirable via PEXGetEnumTypeInfo(3).

If one or more primitives were picked, a pick status of PEXPick is returned along with the pick paths. The hierarchical pick path is equivalent to the renderer’s current path at the time the picked primitive was processed. If no primitives were picked, the returned pick status is PEXNoPick, and the returned pick path is a null pointer. If the renderer’s drawable was destroyed or resized during the pick operation, the returned pick status is PEXAbortedPick and the returned pick path is a null pointer.

The paths of all hit primitives are recorded until reaching the maximum number of hits or until the server maximum number of recordable hits is reached. Once the maximum number of paths is recorded, subsequent primitives may be ignored and the results returned.

modified November 1995
If all possible hits were recorded, then \texttt{PEXNoMoreHits} is returned and the renderer’s pick start path will be empty. If the maximum number of hits was reached and additional hits were detected, then \texttt{PEXMoreHits} is returned and the renderer’s pick start path will be set to the last recorded hit primitive. If, after reaching the maximum number of hits, subsequent output commands were ignored, then \texttt{PEXMaybeMoreHits} is returned and the renderer’s pick start path will be set to the last element processed by the renderer before it started ignoring primitives.

If the specified drawable does not have the same root and depth as the drawable used to create the renderer, or, if the specified drawable is not one of the supported drawables returned by \texttt{PEXMatchRenderingTargets}, a match error is generated. If the renderer state is set to \texttt{PEXRendering} or \texttt{PEXPicking} when this function is called, then the operation in progress is aborted, the \texttt{PEXPickAll} function is completed, and a \texttt{BadPEXRenderer-State} error returned.

\texttt{PEXlib} allocates memory for the return value. \texttt{PEXFreePickPaths(3)} should be called to deallocate the memory.

**DATA STRUCTURES**

```c
typedef XID PEXRenderer;

typedef union {
    PEXPDNPCHitVolume volume;
    PEXPDDCHitBox box;
    PEXPickDataRecord data;
} PEXPickRecord;

typedef PEXNPCSubVolume PEXPDNPCHitVolume;

typedef struct {
    PEXCoord min;
    PEXCoord max;
} PEXNPCSubVolume;

typedef struct {
    float x;
    float y;
    float z;
} PEXCoord;

typedef struct {
    PEXDeviceCoord2D position;
    float distance;
} PEXPDDCHitBox;

typedef struct {
    short x;
    short y;
}```
typedef struct {
    unsigned short length; /* number of bytes in record */
    char *record;
} PEXPickDataRecord;

typedef struct {
    unsigned long count; /* number of elements */
    PEXPickElementRef *elements;
} PEXPickPath;

typedef struct {
    PEXStructure sid;
    unsigned long offset;
    unsigned long pick_id;
} PEXPickElementRef;

typedef XID PEXStructure;

ERRORS

BadAlloc
The server failed to allocate resources necessary to complete request.

BadDrawable
The specified drawable resource identifier is invalid.

BadMatch
The specified drawable is unsupported, or the specified renderer resource was not created with a compatible drawable.

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXRendererState
The specified renderer was in an invalid state.

BadPEXPath
The renderer pick start path is invalid.

BadValue
The pick record contains invalid data, or the pick device type is invalid.

SEE ALSO
PEXBeginPickAll(3)
PEXEndPickAll(3)
PEXFreePickPaths(3)
NAME
PEXPickOne - Pick One Traversal

SYNTAX
PEXPickPath *PEXPickOne(Display *display, Drawable drawable, PEXRenderer renderer, 
PEXStructure structure, int method, int pick_device_type, PEXPickRecord *pick_record, int *status_return, int *undetectable_return)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
drawable The resource identifier of a drawable.
renderer The resource identifier of the renderer.
structure The resource identifier for the root structure of the structure network.
method The pick one method (PEXPickLast, PEXPickClosestZ, PEXPickVisibleAny, PEXPickVisibleClosest).
pick_device_type The pick device type (PEXPickDeviceDCHitBox or PEXPickDeviceNPCHitVolume).
pick_record A pointer to the pick data record.
status_return Returns the status of the pick operation.
undetectable_return Returns True or False indicating whether another pick better satisfied the pick criteria with the exception that it did not pass the pick filter test.

RETURNS A pointer to the pick path; a null pointer if unsuccessful or no pick (see also status_return).

DESCRIPTION This function traverses the specified structure network.
Standard pick one methods are PEXPickLast, PEXPickClosestZ, PEXPickVisibleAny and PEXPickVisibleClosest. The supported pick device types are inquirable via PEXGetEnumTypeInfo(3).
If a primitive was picked, the returned pick status is PEXPick. If no primitive was picked, the returned pick status is PEXNoPick, and the returned pick path is a null pointer. If the renderer's drawable was destroyed or resized during the pick operation, the returned pick status is PEXAbortedPick and the returned pick path is a null pointer.
If there was a primitive which more closely satisfied the pick criteria, but did not pass the pick filter test, then the undetectable pick return status will be True. Otherwise, it will be False.
If the specified drawable does not have the same root and depth as the drawable that was used to create the renderer, or, if the specified drawable is not one of the supported drawables returned by PEXMatchRenderingTargets(3), a Match error will be generated.
If the renderer state is set to PEXRendering or PEXPicking when this function is called, then the operation in progress is aborted, the PEXPickOne function is completed, and a

modified November 1995
BadPEXRendererState error is sent. PEXlib allocates memory for the return value. PEXFreePickPaths(3) should be called to deallocate the memory.

typedef XID PEXRenderer;

typedef union {
  PEXPDPNPCHitVolume volume;
  PEXPDDCHitBox box;
  PEXPickDataRecord data;
} PEXPickRecord;

typedef PEXNPCSubVolume PEXPDNPCHitVolume;

typedef struct {
  PEXCoord min;
  PEXCoord max;
} PEXNPCSubVolume;

typedef struct {
  float x;
  float y;
  float z;
} PEXCoord;

typedef struct {
  PEXDeviceCoord2D position;
  float distance;
} PEXPDDCHitBox;

typedef struct {
  short x;
  short y;
} PEXDeviceCoord2D;

typedef struct {
  unsigned short length; /* number of bytes in record */
  char *record;
} PEXPickDataRecord;

typedef struct {
  unsigned long count; /* number of elements */
  PEXPickElementRef *elements;
} PEXPickPath;
typedef struct {
    PEXStructure sid;
    unsigned long offset;
    unsigned long pick_id;
} PEXPickElementRef;

typedef XID PEXStructure;

ERRORS

BadAlloc
The server failed to allocate resources necessary to complete request.

BadDrawable
The specified drawable resource identifier is invalid.

BadMatch
The specified drawable is unsupported, or the specified renderer resource was not created with a compatible drawable.

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXRendererState
The specified renderer was in an invalid state.

BadPEXStructure
The specified structure resource identifier is invalid.

BadValue
The pick record contains invalid data, or the pick device type is invalid.

SEE ALSO
PEXBeginPickOne(3)
PEXEndPickOne(3)
NAME
PEXPolarViewMatrix - utility function

SYNTAX
int PEXPolarViewMatrix(PEXCoord *from, double distance, double azimuth, double altitude, double twist, PEXMatrix matrix_return)

PARAMETERS
from The viewing position.
distance The distance between the from position and the position being viewed.
azimuth The angle in the x,z plane from the +z axis to the line of sight, in radians. Positive values are counter-clockwise when viewed from the positive y axis.
altitude The angular inclination of the line of sight from the x,z plane. The altitude argument is the angle in radians. Positive values are towards the positive y axis.
twist The up direction of the view, given as an angle, in radians, about the line of sight. Positive values of twist are in the counter-clockwise direction.
matrix_return Matrix in which result is stored.

RETURNS
Zero if successful; otherwise, one of the following:
PEXBadDistance

DESCRIPTION
This routine formats a polar view orientation matrix. This routine is similar to PEXLookAtViewMatrix(3), except that the viewing parameters are specified in spherical coordinates. The from position defines one end of the view plane normal; the position indicated by distance, azimuth, and altitude define the base of the view plane normal, and the origin of VRC.

The view is defined with respect to the from position (the viewing position) and the distance between it and the position being viewed. The azimuth angle specifies the direction of the line of sight going toward the position being viewed. Positive values of azimuth are counter-clockwise when viewed from the positive y axis.

The azimuth and altitude angles apply to the coordinate system with from at the origin and the line of sight emanating from it. The azimuth specifies the angle between the line of sight and the +z axis, and the altitude defines the angle between it and the x,z plane.

When applied, the transformation places the viewing position at the origin, aligns the viewpoint with the +z axis, applies any twist to the coordinates, and then places the viewed point at the origin.

ERRORS
None

SEE ALSO
PEXLookAtViewMatrix(3)
PEXViewOrientationMatrix(3)
PEXViewMappingMatrix(3)
### NAME
PEXPolyline - 3D Polyline Primitive

### SYNTAX
void PEXPolyline(Display *display, XID resource_id, PEXOCRequestType req_type, unsigned int count, PEXCoord *points)

### PARAMETERS
- **display**: A pointer to a display structure returned by a successful XOpenDisplay call.
- **resource_id**: The resource identifier of the renderer or structure.
- **req_type**: The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRendersingle or PEXOCStoresingle).
- **count**: The number of points.
- **points**: An array of points defining the polyline.

### RETURNS
None

### DESCRIPTION
This function creates a polyline output primitive. The points, specified in modeling coordinates, are joined together by line segments. The first point is joined to the second, the second to the third, and so on. The last point is not joined to the first.

The polyline colors are affected only by depth-cueing and are mapped to device colors. Polylines are not displayed if they are outside the currently defined clipping volume. Polylines crossing the clipping volume are clipped and only the portions inside the clipping volume are displayed.

Depending on the ASF attributes, the line color, line type, line width, and polyline interpolation method attributes are obtained either directly from the current attributes or from the line bundle.

A polyline with fewer than two points is considered degenerate. It is stored in a structure, but when rendered, the primitive is ignored and has no visual effect.

### DATA STRUCTURES
See PEXlib.h.

### ERRORS
- **BadPEXRenderer**: The specified renderer resource identifier is invalid.
- **BadPEXStructure**: The specified structure resource identifier is invalid.

### SEE ALSO
- PEXSetLineType(3)
- PEXSetLineWidth(3)
- PEXSetLineColorIndex(3)
- PEXSetLineColor(3)
- PEXSetLineBundleIndex(3)
NAME
PEXPolyline2D - 2D Polyline Primitive

SYNTAX
void PEXPolyline2D(Display *display, XID resource_id, PEXOCRequestType req_type, unsigned int count, PEXCoord2D *points)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
count The number of points.
points An array of points defining the polyline.

RETURNS None

DESCRIPTION
This function creates a 2D polyline output primitive.
This function is like PEXPolyline(3), except that the points consist of only x- and y-components. The z-component is assumed to be zero. This primitive is two-dimensional only in that the z-components are implied. Geometry transformations are still carried out in three dimensions.

DATA STRUCTURES
See PEXlib.h.

ERRORS
BadPEXRender
The specified renderer resource identifier is invalid.
BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO
PEXSetLineType(3)
PEXSetLineWidth(3)
PEXSetLineColorIndex(3)
PEXSetLineColor(3)
PEXSetLineBundleIndex(3)
**NAME**
PEXPolylineSetWithData - 3D Set of Polylines Primitive

**SYNTAX**
void PEXPolylineSetWithData(Display *display, XID resource_id, PEXOCRequestType req_type, unsigned int vertex_attributes, int color_type, unsigned int count, PEXListOfVertex *vertex_lists)

**PARAMETERS**
- **display**
  A pointer to a display structure returned by a successful XOpenDisplay call.
- **resource_id**
  The resource identifier of the renderer or structure.
- **req_type**
  The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- **vertex_attributes**
  A mask indicating the vertex attributes provided (PEXGANone or PEXGAColor).
- **color_type**
  The type of color data provided (PEXColorTypeIndexed, PEXColorTypeRGB, PEXColorTypeCIE, PEXColorTypeHSV, PEXColorTypeHLS, PEXColorTypeRGB8, PEXColorTypeRGB16).
- **count**
  The number of polylines.
- **vertex_lists**
  A pointer to a list of vertex arrays defining each polyline in the set.

**RETURNS**
None

**DESCRIPTION**
This function creates a polyline set output primitive.

This function is similar to PEXPolyline(3) except that it allows applications to specify a number of polylines as well as additional data. These polylines may contain a color value at each vertex. The vertex attributes indicate the content of the additional data. Color values passed must be of the specified color type.

If colors are passed for each vertex, they are used instead of the line color attribute. The polyline interpolation method, which depending on the ASF attribute is obtained directly from the current polyline interpolation attribute or from the line bundle, controls how the polylines are shaded.

All other aspects of this primitive are the same as PEXPolyline(3).

**DATA STRUCTURES**
See PEXlib.h.

**ERRORS**
- **BadPEXOutputCommand**
  The output command contains an invalid value.
- **BadPEXRenderer**
  The specified renderer resource identifier is invalid.
- **BadPEXStructure**
  The specified structure resource identifier is invalid.
SEE ALSO

PEXSetLineType(3)
PEXSetLineWidth(3)
PEXSetLineColorIndex(3)
PEXSetLineColor(3)
PEXSetPolylineInterpMethod(3)
PEXSetLineBundleIndex(3)

modified November 1995
NAME PEXPostStructure - Post Structure to a Workstation

SYNTAX void PEXPostStructure(Display *display, PEXWorkstation workstation, PEXStructure structure, double priority)

PARAMETERS display A pointer to a display structure returned by a successful XOpenDisplay call.
workstation The resource identifier of the workstation.
structure The resource identifier of the structure.
priority The priority of the newly-posted structure.

RETURNS None

DESCRIPTION This function adds the specified structure to the list of posted structures in the specified workstation. The priority is used to indicate the priority of the new posted structure relative to the structures already posted. If multiple structures are posted for display to the same display space location, the higher priority structure will be displayed. If two structures have the same priority, the last posted structure is considered of higher priority.

DATA STRUCTURES
typedef XID PEXStructure;
typedef XID PEXWorkstation;

ERRORS BadPEXStructure
The specified structure resource identifier is invalid.
BadPEXWorkstation
The specified workstation resource identifier is invalid.

SEE ALSO PEXUnpostStructure(3)
PEXUnpostAllStructures(3)

modified November 1995
NAME
PEXQuadrilateralMesh - 3D Quadrilateral Mesh Primitive

SYNTAX
void PEXQuadrilateralMesh(Display *display, XID resource_id, PEXOCRequestType
  req_type, int shape_hint, unsigned int facet_attributes, unsigned int
  vertex_attributes, int color_type, PEXArrayOfFacetData facet_data, unsigned int
  col_count, unsigned int row_count, PEXArrayOfVertex vertices)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay
call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOC-
Store, PEXOCRenderSingle or PEXOCStoreSingle).
shape_hint The shape which describes all of the quadrilaterals (PEXShapeComplex,
PEXShapeNonConvex, PEXShapeConvex, PEXShapeUnknown).
facet_attributes A mask indicating the facet attributes provided (PEXGANone, PEX-
GAColor, PEXGANormal).
vertex_attributes A mask indicating the vertex attributes provided (PEXGANone, PEX-
GAColor, PEXGANormal).
color_type The type of color data provided (PEXColorTypeIndexed, PEXColor-
TypeRGB, PEXColorTypeCIE, PEXColorTypeHSV, PEXColor-
TypeHLS, PEXColorTypeRGB8, PEXColorTypeRGB16).
facet_data An array of facet data.
col_count The number of columns in the vertex array.
row_count The number of rows in the vertex array.
vertices A two-dimensional (row-major) array of vertices defining the quadrila-
teral mesh.

RETURNS None

DESCRIPTION
This function creates a quadrilateral mesh output primitive.
The quadrilateral mesh surface is created from the vertices. The vertex array is accessed
in row major order (i.e. the column number varies fastest as vertices are accessed). The
(i, j), (i+1, j), (i+1, j+1), and (i, j+1) vertices are connected to create a single
facet. Adjacent vertices are interconnected until the entire facet network is processed.
Normals for quadrilaterals, if not provided explicitly, are computed by taking the cross
product of the diagonals. For a quadrilateral with the above vertices, the cross product
would be formed as follows:

  normal(i,j) = (point(i+1, j+1) - point(i, j)) X (point(i, j+1) - point(i+1, j))

Normals are assumed to be unit length vectors. If not unit length, the result is
implementation-dependent.
There must be an entry in the facet data array for each facet, if facet data is indicated by the facet attributes.

All other aspects of this primitive are the same as `PEXFillAreaSetWithData(3)`.

### DATA STRUCTURES

See `PEXlib.h`.

### ERRORS

- **BadPEXOutputCommand**
  - The output command contains an invalid value.

- **BadPEXRenderer**
  - The specified renderer resource identifier is invalid.

- **BadPEXStructure**
  - The specified structure resource identifier is invalid.

### SEE ALSO

- `PEXSetInteriorStyle(3)`
- `PEXSetInteriorStyleIndex(3)`
- `PEXSetSurfaceColorIndex(3)`
- `PEXSetSurfaceColor(3)`
- `PEXSetReflectionAttributes(3)`
- `PEXSetReflectionModel(3)`
- `PEXSetSurfaceInterpMethod(3)`
- `PEXSetBFInteriorStyle(3)`
- `PEXSetBFInteriorStyleIndex(3)`
- `PEXSetBFSurfaceColorIndex(3)`
- `PEXSetBFSurfaceColor(3)`
- `PEXSetBFReflectionAttributes(3)`
- `PEXSetBFReflectionModel(3)`
- `PEXSetBFSurfaceInterpMethod(3)`
- `PEXSetFacetCullingMode(3)`
- `PEXSetFacetDistinguishFlag(3)`
- `PEXSetPatternSize(3)`
- `PEXSetPatternAttributes(3)`
- `PEXSetPatternAttributes2D(3)`
- `PEXSetInteriorBundleIndex(3)`
- `PEXSetSurfaceEdgeFlag(3)`
- `PEXSetSurfaceEdgeType(3)`
- `PEXSetSurfaceEdgeWidth(3)`
- `PEXSetSurfaceEdgeColor(3)`
- `PEXSetSurfaceEdgeColorIndex(3)`
- `PEXSetEdgeBundleIndex(3)`
NAME  PEXQueryEncodedTextExtents - Query Encoded Text Extents

SYNTAX  PEXTExtText *PEXQueryEncodedTextExtents(Display *display, XID resource_id, unsigned int font_table_index, int path, double expansion, double spacing, double height, int halign, int valign, unsigned long count, PEXListOfEncodedText *encoded_text)

PARAMETERS  

**display**  A pointer to a display structure returned by a successful XOpenDisplay call.

**resource_id**  The resource identifier of either a renderer, a workstation or a text font table.

**font_table_index**  The text font table index.

**path**  The text path (PEXPathRight, PEXPathLeft, PEXPathUp, PEXPathDown).

**expansion**  The text character expansion factor.

**spacing**  The text character spacing factor.

**height**  The text character height.

**halign**  The text horizontal text alignment (PEXHAlignNormal, PEXHAlignLeft, PEXHAlignCenter, PEXHAlignRight).

**valign**  The text vertical text alignment (PEXVAlignNormal, PEXVAlignTop, PEXVAlignCap, PEXVAlignHalf, PEXVAlignBase, PEXVAlignBottom).

**count**  The number of encoded text strings.

**encoded_text**  An array of encoded text strings.

RETURNS  An array of text extents; a null pointer if unsuccessful.

DESCRIPTION  This function is like PEXQueryTextExtents(3), except that multiple encoded text strings are specified. Each text string in the encoded text list has a character set, a character set width, an encoding state, and a list of characters. (See PEXEncodedTextData.)

The character set is an index into the current font group. Font groups have individual fonts which are numbered starting at one; a value of three would select the third font in the font group currently being used. If a character set is not available in the current font group then the default font group is used. If a character set value is not available in the default font group then the result is implementation-dependent. The character set width indicates the width or size of characters in the strings. Valid values for character set width are PEXCSByte, PEXCSShort and PEXCSLong. The encoding state is provided for use by the application and is not interpreted by the PEX server.

All other aspects of this function are the same as PEXQueryTextExtents(3).
typedef struct {
    unsigned short count; /* number of encodings */
    PEXEncodedTextData *encoded_text;
} PEXListOfEncodedText;

typedef struct {
    unsigned short character_set;
    unsigned char character_set_width;
    unsigned char encoding_state;
    unsigned short reserved;
    unsigned short length;
    char *ch;
} PEXEncodedTextData;

typedef struct {
    PEXCoord2D lower_left;
    PEXCoord2D upper_right;
    PEXCoord2D concat_point;
} PEXTextExtent;

typedef struct {
    float x;
    float y;
} PEXCoord2D;

ERRORS

BadMatch
The specified resource identifier identifies a table of type other than a text font table.

BadValue
A specified value for one or more text attributes is invalid, or the specified resource identifier does not identify a valid renderer, workstation or lookup table resource.

SEE ALSO PEXQueryTextExtents(3)
NAME
PEXQueryFont - Query PEX Font Information

SYNTAX
PEXFontInfo *PEXQueryFont(Display *display, PEXFont font)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
font The resource identifier of the font.

RETURNS
A pointer to the font info structure; a null pointer if unsuccessful.

DESCRIPTION
This function returns information about the specified PEX font. The information returned is identical to what PEXListFontsWithInfo(3) would return. PEXlib allocates memory for the returned list of font information. PEXFreeFontInfo(3) should be called to deallocate the memory.

DATA STRUCTURES
typedef XID PEXFont;
typedef struct {
    unsigned long first_glyph;
    unsigned long last_glyph;
    unsigned long default_glyph;
    Bool all_exist;
    Bool stroke;
    unsigned short count; /* number of properties */
    PEXFontProp *props;
} PEXFontInfo;
typedef struct {
    Atom name;
    unsigned long value;
} PEXFontProp;

ERRORS
BadPEXFont
The specified font resource identifier is invalid.

modified November 1995
**NAME**
PEXQueryTextExtents - Query Text Extents

**SYNTAX**
PEXTextExtent *PEXQueryTextExtents(Display *display, XID resource_id, unsigned int font_table_index, int path, double expansion, double spacing, double height, int halign, int valign, unsigned long count, PEXStringData *text)

**PARAMETERS**
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of either a renderer, a workstation or a text font table.
font_table_index The text font table index.
path The text path (PEXPathRight, PEXPathLeft, PEXPathUp, PEXPathDown).
expansion The text character expansion factor.
spacing The text character spacing factor.
height The text character height.
halign The text horizontal text alignment (PEXHAlignNormal, PEXHAlignLeft, PEXHAlignCenter, PEXHAlignRight).
valign The text vertical text alignment (PEXVAlignNormal, PEXVAlignTop, PEXVAlignCap, PEXVAlignHalf, PEXVAlignBase, PEXVAlignBottom).
count The number of text strings.
text An array of text strings.

**RETURNS**
An array of text extents; a null pointer if unsuccessful.

**DESCRIPTION**
This function returns the text extents, in the local 2D text coordinate system, of each specified text string. If the resource identifier is a renderer or PHIGS workstation, then the text font table associated with the renderer or workstation is used. If the resource identifier is a text font lookup table, it is used directly. The specified font group provides the index of the entry that is to be used to obtain the font group. The first character set in the font will be used.

Stroke precision is assumed. The text position is (0,0) in the local 2D text coordinate system. The extent of each string is computed independent of the other strings.

If a specified font has no defined default glyph, then undefined glyphs are taken to have all zero metrics.

The extent data is returned in memory allocated by PEXlib. XFree should be called to deallocate the memory.
typedef struct {
    unsigned short length;
    char *ch;
} PEXStringData;

typedef struct {
    PEXCoord2D lower_left;
    PEXCoord2D upper_right;
    PEXCoord2D concat_point;
} PEXTextExtent;

typedef struct {
    float x;
    float y;
} PEXCoord2D;

**ERRORS**

**BadMatch**
The specified resource identifier identifies a table of type other than a text font table.

**BadValue**
A specified value for one or more text attributes is invalid, or the specified resource identifier does not identify a valid renderer, workstation or lookup table resource.

**SEE ALSO**
PEXQueryEncodedTextExtents(3)
### NAME
PEXRedrawAllStructures - Redraw All Posted Structures

<table>
<thead>
<tr>
<th>SYNTAX</th>
<th>void PEXRedrawAllStructures(Display *display, PEXWorkstation workstation)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>PARAMETERS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>display</td>
<td>A pointer to a display structure returned by a successful XOpenDisplay call.</td>
</tr>
<tr>
<td>workstation</td>
<td>The resource identifier of the workstation.</td>
</tr>
</tbody>
</table>

| RETURNS               | None                                                                                |

| DESCRIPTION           | This function redraws all of the posted structures for the workstation. First, if the workstation’s display surface is `PEXNotEmpty`, its drawable is cleared to the color stored in entry zero of the color table. Then, if any of its view, workstation, HLHSR or buffer update attributes is set to `PEXPending`, the requested values are made current and the update attributes are set to `PEXNotPending`. After this, all the posted structures are traversed and rendered (in priority order). Finally, the workstation’s visual state is set to `PEXCorrect`. |

<table>
<thead>
<tr>
<th>DATA STRUCTURES</th>
<th>typedef XID PEXWorkstation;</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>ERRORS</th>
<th>BadPEXWorkstation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The specified workstation resource identifier is invalid.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SEE ALSO</th>
<th>PEXUpdateWorkstation(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PEXExecuteDeferredActions(3)</td>
</tr>
</tbody>
</table>
NAME  PEXRedrawClipRegion - Redraw Posted Structures Clipped to Clip Region

SYNTAX  void PEXRedrawClipRegion(Display ∗display, PEXWorkstation workstation, unsigned long count, PEXDeviceRect ∗clip_rectangles)

PARAMETERS  
  display  A pointer to a display structure returned by a successful XOpenDisplay call.
  workstation  The resource identifier of the workstation.
  count  The number of clip rectangles.
  clip_rectangles  An array of device clip rectangles.

RETURNS  None

DESCRIPTION  This function performs actions similar to PEXRedrawAllStructures(3) except that the workstation state attributes are not modified or updated by this function and rendering will occur only in the region defined by the specified clip list. The color stored in entry zero of the color table is used to clear the region defined by the clip list. The clip list must consist of non-overlapping rectangles or the result will be undefined. If a z-buffering HLHSR algorithm is used, only the z-values corresponding to pixels in the clip region will be affected. All of the posted structures for the workstation are redrawn, but clipped to the clip list. Pending changes are not made current, nor is the visual state modified.

DATA STRUCTURES
  typedef XID PEXWorkstation;

  typedef struct {
    short xmin;
    short ymin;
    short xmax;
    short ymax;
  } PEXDeviceRect;

ERRORS  BadPEXWorkstation
  The specified workstation resource identifier is invalid.

SEE ALSO  PEXRedrawAllStructures(3)
  PEXExecuteDeferredActions(3)
| **NAME** | PEXRemoveFromNameSet - Remove Names from Name Set |
| **SYNTAX** | void PEXRemoveFromNameSet(Display *display, XID resource_id, PEXOCRequestType req_type, unsigned long count, PEXName *names) |
| **PARAMETERS** | display | A pointer to a display structure returned by a successful XOpenDisplay call. |
| | resource_id | The resource identifier of the renderer or structure. |
| | req_type | The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle). |
| | count | The number of names. |
| | names | An array of names to be removed from the current name set. |
| **RETURNS** | None |
| **DESCRIPTION** | This function creates an output primitive attribute which removes the specified list of names from the current name set. If any name is outside the supported range, that name is ignored. |
| **DATA STRUCTURES** | typedef unsigned long PEXName; |
| **ERRORS** | BadPEXRenderer | The specified renderer resource identifier is invalid. |
| | BadPEXStructure | The specified structure resource identifier is invalid. |
| **SEE ALSO** | PEXCreateNameSet(3) |
| | PEXAddToNameSet(3) |
| | PEXGetImpDepConstants(3) |

modified November 1995
NAME  PEXRenderElements - Render Elements

SYNTAX  void PEXRenderElements(Display *display, PEXRenderer renderer, PEXStructure structure, int whence1, long offset1, int whence2, long offset2)

PARAMETERS  
  display  A pointer to a display structure returned by a successful XOpenDisplay call.
  renderer  The resource identifier of a renderer resource.
  structure  The resource identifier of a structure resource.
  whence1  A value specifying, with offset1, the first limit of the element range (PEXBeginning, PEXCurrent PEXEnd).
  offset1  An offset from whence1 specifying the first limit of the element range.
  whence2  A value specifying, with offset2, the second limit of the element range (PEXBeginning, PEXCurrent PEXEnd).
  offset2  An offset from whence2 specifying the second limit of the element range.

RETURNS  None

DESCRIPTION  
This function processes all output commands in the specified element range of the specified structure. Output primitives in structure are rendered using the specified renderer. If the renderer is not rendering or picking, the request will be ignored. Structures referenced through execute structure output commands are also processed.

The first limit of the range is defined by whence1 and offset1 and the second limit of the range is defined by whence2 and offset2. The whence values describe how to interpret the corresponding offset. PEXBeginning means the element position is the value of offset (i.e. the offset from the beginning of the structure). PEXCurrent means the element position is the value of the current element pointer position plus the value of offset (i.e. the offset from the current element pointer). PEXEnd means the element position is the value of the last element position in the structure plus the value of offset (i.e. the offset from the end of the structure). Offsets can be negative values. If after computing an element position, it is less than zero, the position will be set to zero. If after computing an element position, it is greater than the number of elements in the structure, it will be set to the last structure element in the structure.

DATA STRUCTURES  
typedef XID PEXRenderer;
typedef XID PEXStructure;

ERRORS  
BadPEXRenderer  The specified renderer resource identifier is invalid.
BadPEXStructure  The specified structure resource identifier is invalid.
BadValue
The specified value for *whence* was invalid.

**SEE ALSO**
- PEXBeginRendering(3)
- PEXCreateStructure(3)
- PEXCreateRenderer(3)
### NAME
PEXRenderNetwork - Render Network

### SYNTAX
```c
void PEXRenderNetwork(Display *display, Drawable drawable, PEXRenderer renderer, PEXStructure structure)
```

### PARAMETERS
- **display**: A pointer to a display structure returned by a successful `XOpenDisplay` call.
- **drawable**: The resource identifier of a drawable.
- **renderer**: The resource identifier of a renderer resource.
- **structure**: The resource identifier of a structure resource.

### RETURNS
None

### DESCRIPTION
This function processes all output commands stored in `structure` using the specified `renderer`. Output primitives in `structure` are rendered to the specified `drawable`. Structures referenced through execute structure output commands are also processed. This request effectively performs an implicit `PEXBeginRendering(3)` before the traversal of the specified structure network and an implicit `PEXEndRendering(3)`, with a flush value of `True`, after the traversal.

### DATA STRUCTURES
```c
typedef XID PEXRenderer;
typedef XID PEXStructure;
```

### ERRORS
- **BadAlloc**: The server failed to allocate resources necessary to complete request.
- **BadDrawable**: The specified drawable resource identifier is invalid.
- **BadMatch**: The specified drawable is unsupported, or the specified renderer resource was not created with a compatible drawable.
- **BadPEXRenderer**: The specified renderer resource identifier is invalid.
- **BadPEXRendererState**: The specified renderer was in an invalid state.
- **BadPEXStructure**: The specified structure resource identifier is invalid.

### SEE ALSO
- `PEXExecuteStructure(3)`
- `PEXBeginRendering(3)`
- `PEXEndRendering(3)`
- `PEXCreateStructure(3)`
- `PEXCreateRenderer(3)`

modified November 1995
PEXRestoreModelClipVolume - Restore Model Clip Volume

**SYNTAX**

```c
void PEXRestoreModelClipVolume(Display *display, XID resource_id, PEXOCRequestType req_type)
```

**PARAMETERS**

- `display` A pointer to a display structure returned by a successful `XOpenDisplay` call.
- `resource_id` The resource identifier of the renderer or structure.
- `req_type` The request type for the output command (`PEXOCRender`, `PEXOCStore`, `PEXOCRenderSingle` or `PEXOCStoreSingle`).

**RETURNS**

None

**DESCRIPTION**

This function creates an output primitive attribute which restores the last-saved modeling clipping volume. If there is no last-saved clipping volume that can be restored, the modeling clipping volume is restored to its default state.

**ERRORS**

- `BadPEXRenderer` The specified renderer resource identifier is invalid.
- `BadPEXStructure` The specified structure resource identifier is invalid.

**SEE ALSO**

PEXSetModelClipFlag(3)  
PEXSetModelClipVolume(3)
NAME  PEXRotate - utility function

SYNTAX  int PEXRotate(int axis, double angle, PEXMatrix matrix_return)

PARAMETERS  
axis  One of PEXXAxis, PEXYAxis, PEXZAxis.
angle  Angle of the rotation in radians.
matrix_return  Matrix into which rotation matrix is stored.

RETURNS  Zero if successful; otherwise, one of the following:
PEXBadAxis  Invalid axis value specified.

DESCRIPTION  This function creates a rotation matrix about the specified axis. The resulting matrix rotates by the angle specified in radians about the origin. The function returns unsuccessfully if axis is not one of the defined values.

ERRORS  None

SEE ALSO  PEXRotate2D(3)  PEXRotateGeneral(3)
NAME  PEXRotate2D - utility function

SYNTAX  void PEXRotate2D(double angle, PEXMatrix3x3 matrix_return)

PARAMETERS  
    angle  Angle of the rotation in radians.
    matrix_return  Matrix into which rotation matrix is stored.

RETURNS  None

DESCRIPTION  The resulting matrix rotates by the angle specified in radians about the origin.

ERRORS  None

SEE ALSO  PEXRotate(3)
        PEXRotateGeneral(3)
**NAME**
PEXRotateGeneral - utility function

**SYNTAX**

```c
int PEXRotateGeneral(PEXCoord *point1, PEXCoord *point2, double angle, PEXMatrix matrix_return)
```

**PARAMETERS**

- `point1` Endpoint of line segment defining rotation axis.
- `point2` Endpoint of line segment defining rotation axis.
- `angle` Angle of the rotation in radians.
- `matrix_return` Matrix into which rotation matrix is stored.

**RETURNS**
Zero if successful; otherwise, one of the following:

- **PEXBadAxis**
  The endpoints are coincident.

**DESCRIPTION**

This routine formats a matrix for a right-handed rotation about an arbitrary axis. The rotation axis is defined by the line segment joining the two points. The `angle` is in radians and is measured counter-clockwise when looking from the first point to second point along the specified axis.

Returns unsuccessfully if the points are coincident.

**ERRORS**

None

**SEE ALSO**

- PEXRotate(3)
- PEXRotate2D(3)

modified November 1995
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXScale - utility function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td><code>void PEXScale(PEXVector *scale_vector, PEXMatrix matrix_return)</code></td>
</tr>
<tr>
<td>PARAMETERS</td>
<td><code>scale_vector</code> Vector containing the X, Y and Z scale factors.</td>
</tr>
<tr>
<td></td>
<td><code>matrix_return</code> Matrix into which rotation matrix is stored.</td>
</tr>
<tr>
<td>RETURNS</td>
<td>None</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>Creates a scale matrix given the scale vector.</td>
</tr>
<tr>
<td>ERRORS</td>
<td>None</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>PEXScale2D(3)</td>
</tr>
<tr>
<td>NAME</td>
<td>PEXScale2D - utility function</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>SYNTAX</td>
<td><code>void PEXScale2D(PEXVector2D *scale_vector, PEXMatrix3x3 matrix_return)</code></td>
</tr>
<tr>
<td>PARAMETERS</td>
<td><code>scale_vector</code> Vector containing the X and Y scale factors.</td>
</tr>
<tr>
<td></td>
<td><code>matrix_return</code> Matrix into which rotation matrix is stored.</td>
</tr>
<tr>
<td>RETURNS</td>
<td>None</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>Creates a 3X3 scale matrix given the scale vector.</td>
</tr>
<tr>
<td>ERRORS</td>
<td>None</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>PEXScale(3)</td>
</tr>
</tbody>
</table>
NAME
PEXSearchNetwork - Search Network

SYNTAX
Status PEXSearchNetwork(Display *display, PEXSearchContext context, PEXStructurePath **path_return)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
context The resource identifier of the search context.
path_return Returns a pointer to a structure network path identifying the first primitive found.

RETURNS Zero if unsuccessful; non-zero otherwise.

DESCRIPTION
This function searches a structure network according to the specified search criteria. The path to the first primitive found that satisfies the search criteria is returned. If no primitive is found that satisfies the criteria, a null pointer is returned.

After the search has been completed, the start path attribute of the specified search context will be set to the path returned, if a primitive was found.

PEXlib allocates memory for the returned structure path. PEXFreeStructurePaths(3) should be called to deallocate the memory.

DATA STRUCTURES
typedef XID PEXSearchContext;

typedef struct {
    unsigned long count; /* number of elements */
    PEXElementRef *elements;
} PEXStructurePath;

typedef struct {
    PEXStructure structure;
    unsigned long offset;
} PEXElementRef;

typedef XID PEXStructure;

ERRORS
BadPEXPath The specified path is invalid.
BadPEXSearchContext The specified search context resource identifier is invalid.

SEE ALSO
PEXCreateSearchContext(3)
PEXChangeSearchContext(3)
PEXFreeStructurePaths(3)
NAME
PEXSendOCs - Send Encoded Output Commands

SYNTAX
void PEXSendOCs(Display *display, XID resource_id, PEXOCRequestType req_type, int float_format, unsigned long oc_count, unsigned int length, char *encoded_ocs)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
float_format The floating point format of the encoded output commands (PEXIEEE_754_32, PEXDEC_F_Floating, PEXIEEE_754_64, PEXDEC_D_Floating).
oc_count The number of encoded output commands.
length The length, in bytes, of the encoded output commands.
encoded_ocs A pointer to the encoded output commands.

RETURNS
None

DESCRIPTION
This function sends encoded output commands to the specified PEX server display. Sending output commands to a structure whose editing mode is PEXStructureReplace is not really useful. The behavior will be unpredictable unless a request type of PEXOCStoreSingle is used. And, if the request type is PEXOCStoreSingle, each output command will simply replace the previous one sent. Applications should ensure that the structure’s editing mode is PEXStructureInsert, when sending multiple output commands. If it is intended to replace multiple elements, the application can delete those elements first, and then insert the new ones.

ERRORS
BadPEXFloatingPointFormat
The specified floating point format is invalid or unsupported.
BadPEXOutputCommand
The output command contains an invalid value.
BadPEXRenderer
The specified renderer resource identifier is invalid.
BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO
PEXDecodeOCs(3)
PEXEncodeOCs(3)

modified November 1995
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXSetATextAlignment - Set Annotation Text Alignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td>void PEXSetATextAlignment(Display *display, XID resource_id, PEXOCRequestType req_type, int halignment, int valignment)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>display A pointer to a display structure returned by a successful XOpenDisplay call.</td>
</tr>
<tr>
<td></td>
<td>resource_id The resource identifier of the renderer or structure.</td>
</tr>
<tr>
<td></td>
<td>req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).</td>
</tr>
<tr>
<td></td>
<td>halignment The horizontal annotation text alignment (PEXHAlignNormal, PEXHAlignLeft, PEXHAlignCenter, PEXHAlignRight).</td>
</tr>
<tr>
<td></td>
<td>valignment The vertical annotation text alignment (PEXVAlignNormal, PEXVAlignTop, PEXVAlignCap, PEXVAlignBase, PEXVAlignBottom).</td>
</tr>
<tr>
<td>RETURNS</td>
<td>None</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>This function creates an output primitive attribute which sets the horizontal and vertical annotation text alignment attributes.</td>
</tr>
<tr>
<td>ERRORS</td>
<td>BadPEXOutputCommand The output command contains an invalid value.</td>
</tr>
<tr>
<td></td>
<td>BadPEXRenderer The specified renderer resource identifier is invalid.</td>
</tr>
<tr>
<td></td>
<td>BadPEXStructure The specified structure resource identifier is invalid.</td>
</tr>
</tbody>
</table>

290 modified November 1995
NAME PEXSetATextHeight - Set Annotation Text Height

SYNTAX
void PEXSetATextHeight(Display ∗display, XID resource_id, PEXOCRequestType req_type, double height)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
height The annotation text character height.

RETURNS None

DESCRIPTION
This function creates an output primitive attribute which sets the annotation text height attribute. If the specified height or the computed width is not supported, the height or width is mapped to the nearest supported annotation character height or width. These values depend on the font files that are in the font groups in the selected font table entry, which in turn depend on which X or PEX font files have been opened. If all scalable and rotatable stroke fonts are open, then a continuous range of character sizes are supported. The height is expressed in normalized projection coordinates.

ERRORS
BadPEXRenderer
The specified renderer resource identifier is invalid.
BadPEXStructure
The specified structure resource identifier is invalid.

modified November 1995
**NAME**  
PEXSetATextPath - Set Annotation Text Path

**SYNTAX**  
void PEXSetATextPath(Display display, XID resource_id, PEXOCRequestType req_type, int path)

**PARAMETERS**  
- *display*: A pointer to a display structure returned by a successful XOpenDisplay call.
- *resource_id*: The resource identifier of the renderer or structure.
- *req_type*: The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- *path*: The text (drawing) path (PEXPathRight, PEXPathLeft, PEXPathUp, PEXPathDown).

**RETURNS**  
None

**DESCRIPTION**  
This function creates an output primitive attribute which sets the annotation text *path* attribute.

**ERRORS**  
- **BadPEXOutputCommand**  
  The output command contains an invalid value.
- **BadPEXRenderer**  
  The specified renderer resource identifier is invalid.
- **BadPEXStructure**  
  The specified structure resource identifier is invalid.
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXSetATextStyle - Set Annotation Text Style</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td>void PEXSetATextStyle(Display *display, XID resource_id, PEXOCRequestType req_type, int style)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>display</td>
</tr>
<tr>
<td></td>
<td>resource_id</td>
</tr>
<tr>
<td></td>
<td>req_type</td>
</tr>
<tr>
<td></td>
<td>style</td>
</tr>
<tr>
<td>RETURNS</td>
<td>None</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>This function creates an output primitive attribute which sets the annotation text style attribute. If the specified style is not supported, PEXATextNotConnected is used. Supported values for annotation text style are inquirable via PEXGetEnumTypeInfo(3).</td>
</tr>
<tr>
<td>ERRORS</td>
<td>BadPEXRenderer</td>
</tr>
<tr>
<td></td>
<td>BadPEXStructure</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>PEXGetEnumTypeInfo(3)</td>
</tr>
</tbody>
</table>
NAME  PEXSetATextUpVector - Set Annotation Text Up Vector

SYNTAX  void PEXSetATextUpVector(Display *display, XID resource_id, PEXOCRequestType req_type, PEXVector2D *vector)

PARAMETERS  
  
  *display  A pointer to a display structure returned by a successful XOpenDisplay call.
  *resource_id  The resource identifier of the renderer or structure.
  *req_type  The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
  *vector  The annotation text character up vector.

RETURNS  None

DESCRIPTION  This function creates an output primitive attribute which sets the annotation text up vector attribute. The vector is specified in the text local coordinate system defined by the direction vectors associated with the annotation text primitive. Only the direction of the vector is used, not the magnitude. However, if the annotation text up vector is degenerate (it has a length of zero), a value <0,1> is used.

DATA STRUCTURES  See PEXlib.h.

ERRORS  
  
  *BadPEXRenderer  The specified renderer resource identifier is invalid.
  *BadPEXStructure  The specified structure resource identifier is invalid.
### NAME
PEXSetBFInteriorStyle - Set Back-Face Surface Interior Style

### SYNTAX
```c
void PEXSetBFInteriorStyle(Display *display, XID resource_id, PEXOCRequestType req_type, int style);
```

### PARAMETERS
- **display**: A pointer to a display structure returned by a successful `XOpenDisplay` call.
- **resource_id**: The resource identifier of the renderer or structure.
- **req_type**: The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- **style**: The back-facing surface interior style (PEXInteriorStyleHollow, PEXInteriorStyleSolid, PEXInteriorStylePattern, PEXInteriorStyleHatch, PEXInteriorStyleEmpty).

### RETURNS
None

### DESCRIPTION
This function creates an output primitive attribute which sets the back-facing surface interior style attribute. If the specified style is not supported, `PEXInteriorStyleHollow` is used. Supported values for back-facing surface interior style are inquirable via `PEXGetEnumTypeInfo(3)`.

### ERRORS
- **BadPEXOutputCommand**: The output command contains an invalid value.
- **BadPEXRenderer**: The specified renderer resource identifier is invalid.
- **BadPEXStructure**: The specified structure resource identifier is invalid.

### SEE ALSO
- `PEXGetEnumTypeInfo(3)`
- `PEXSetIndividualASF(3)`

---

modified November 1995 295
NAME

PEXSetBFInteriorStyleIndex - Set Back-Face Surface Interior Style Index

SYNTAX

void PEXSetBFInteriorStyleIndex(Display *display, XID resource_id, PEXOCRequestType req_type, int index)

PARAMETERS

display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
index The back-facing surface interior style index.

RETURNS

None

DESCRIPTION

This function creates an output primitive attribute which sets the back-facing interior style index attribute. If the current back-facing interior style is PEXInteriorStylePattern or PEXInteriorStyleHatch, the specified index is used to further define the rendering style of back-facing surface primitives. For PEXInteriorStylePattern, if the specified pattern table index is not defined, table index one is used. For PEXInteriorStyleHatch, the index determines the hatch style. If the specified hatch style is not supported, style one is used. If style one is not supported, the result is implementation-dependent. Supported values for hatch style are inquirable via PEXGetEnumTypeInfo(3).

ERRORS

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO

PEXGetEnumTypeInfo(3)
PEXSetIndividualASF(3)
PEXCreateLookupTable(3)

modified November 1995
**NAME**
PEXSetBFReflectionAttributes - Set Back-Face Surface Reflection Attributes

**SYNTAX**
void PEXSetBFReflectionAttributes(Display *display, XID resource_id, PEXOCRequestType req_type, PEXReflectionAttributes *attributes)

**PARAMETERS**
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
attributes The back-facing surface reflection attributes.

**RETURNS**
None

**DESCRIPTION**
This function creates an output primitive attribute which sets the back-facing surface reflection attributes. Surface reflection attributes consist of the ambient, diffuse and specular coefficients, the specular concentration and color, and the transmission coefficient.

**DATA STRUCTURES**
typedef struct {
  float ambient;
  float diffuse;
  float specular;
  float specular_conc;
  float transmission;
  PEXColorSpecifier specular_color;
} PEXReflectionAttributes;

See also PEXlib.h.

**ERRORS**
BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

**SEE ALSO**
PEXSetIndividualASF(3)
PEXGetImpDepConstants(3)
<table>
<thead>
<tr>
<th><strong>NAME</strong></th>
<th>PEXSetBFReflectionModel - Set Back-Face Surface Reflection Model</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SYNTAX</strong></td>
<td>void PEXSetBFReflectionModel(Display *display, XID resource_id, PEXOCRequestType req_type, int model)</td>
</tr>
<tr>
<td><strong>PARAMETERS</strong></td>
<td>display A pointer to a display structure returned by a successful XOpenDisplay call.</td>
</tr>
<tr>
<td></td>
<td>resource_id The resource identifier of the renderer or structure.</td>
</tr>
<tr>
<td></td>
<td>req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).</td>
</tr>
<tr>
<td></td>
<td>model The back-facing surface reflection model (PEXReflectionNone, PEXReflectionAmbient, PEXReflectionDiffuse, PEXReflectionSpecular).</td>
</tr>
<tr>
<td><strong>RETURNS</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>DESCRIPTION</strong></td>
<td>This function creates an output primitive attribute which sets the back-facing surface reflection model attribute. If the specified reflection model is not supported, PEXReflectionNone is used. Supported values for back-facing surface reflection model are inquirable via PEXGetEnumTypeInfo(3).</td>
</tr>
<tr>
<td><strong>ERRORS</strong></td>
<td>BadPEXRenderer The specified renderer resource identifier is invalid.</td>
</tr>
<tr>
<td></td>
<td>BadPEXStructure The specified structure resource identifier is invalid.</td>
</tr>
<tr>
<td><strong>SEE ALSO</strong></td>
<td>PEXGetEnumTypeInfo(3)</td>
</tr>
<tr>
<td></td>
<td>PEXSetIndividualASF(3)</td>
</tr>
</tbody>
</table>

modified November 1995
### NAME
PEXSetBFSurfaceColor - Set Back-Face Surface Color

### SYNTAX
```c
void PEXSetBFSurfaceColor(Display *display, XID resource_id, PEXOCRequestType req_type, int color_type, PEXColor *color)
```

### PARAMETERS
- **display**
  A pointer to a display structure returned by a successful `XOpenDisplay` call.
- **resource_id**
  The resource identifier of the renderer or structure.
- **req_type**
  The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- **color_type**
  The type of color (PEXColorTypeIndexed, PEXColorTypeRGB, PEXColorTypeCIE, PEXColorTypeHSV, PEXColorTypeHLS, PEXColorTypeRGB8, PEXColorTypeRGB16).
- **color**
  A pointer to the back-facing surface color.

### RETURNS
None

### DESCRIPTION
This function creates an output primitive attribute which sets the back-facing surface color attribute. The attribute is set to either an indexed color or a direct color value, depending on the color type. If the color type is PEXColorTypeIndexed and the specified color index is not defined, color index one is used. If color index one is not defined, the result is implementation-dependent.

### DATA STRUCTURES
See PEXlib.h.

### ERRORS
- **BadPEXOutputCommand**
  The output command contains an invalid value, e.g. the color type is PEXColorTypeIndexed, and the color index value exceeds 65534.
- **BadPEXRenderer**
  The specified renderer resource identifier is invalid.
- **BadPEXStructure**
  The specified structure resource identifier is invalid.

### SEE ALSO
PEXSetIndividualASF(3)
PEXCreateLookupTable(3)
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXSetBFSurfaceColorIndex - Set Back-Face Surface Color Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td>void PEXSetBFSurfaceColorIndex( Display *display, XID resource_id, PEXOCRequestType req_type, unsigned int index)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>display A pointer to a display structure returned by a successful XOpenDisplay call.</td>
</tr>
<tr>
<td></td>
<td>resource_id The resource identifier of the renderer or structure.</td>
</tr>
<tr>
<td></td>
<td>req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle ).</td>
</tr>
<tr>
<td></td>
<td>index The color table index for back-facing surfaces.</td>
</tr>
<tr>
<td>RETURNS</td>
<td>None</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>This function creates an output primitive attribute which sets the back-facing surface color attribute to an indexed color with the value indicated by index. If the color index is not defined, color index one is used. If color index one is not defined, the result is implementation-dependent.</td>
</tr>
</tbody>
</table>
| ERRORS       | BadPEXOutputCommand  
The color index value exceeds 65534. |
|              | BadPEXRenderer  
The specified renderer resource identifier is invalid. |
|              | BadPEXStructure  
The specified structure resource identifier is invalid. |
| SEE ALSO     | PEXSetIndividualASF(3) |
|              | PEXCreateLookupTable(3) |
NAME  PEXSetBFSurfaceInterpMethod - Set Back-Face Surface Interpolation Method

SYNTAX  void PEXSetBFSurfaceInterpMethod(Display *display, XID resource_id, PEXOCRequestType req_type, int method)

PARAMETERS  display  A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id  The resource identifier of the renderer or structure.
req_type  The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
method  The back-facing surface interpolation method (PEXSurfaceInterpNone, PEXSurfaceInterpColor, PEXSurfaceInterpDotProduct, PEXSurfaceInterpNormal).

RETURNS  None

DESCRIPTION  This function creates an output primitive attribute which sets the back-facing surface interpolation method. If the specified interpolation method is not supported, PEXSurfaceInterpNone is used. Supported values for back-facing surface interpolation method are inquirable via PEXGetEnumTypeInfo(3).

ERRORS  BadPEXRenderer
The specified renderer resource identifier is invalid.
BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO  PEXGetEnumTypeInfo(3)
PEXSetIndividualASF(3)
**NAME**  
PEXSetCharExpansion - Set Character Expansion Factor

**SYNTAX**  
void PEXSetCharExpansion(Display *display, XID resource_id, PEXOCRequestType req_type, double expansion)

**PARAMETERS**  
- **display**: A pointer to a display structure returned by a successful XOpenDisplay call.
- **resource_id**: The resource identifier of the renderer or structure.
- **req_type**: The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- **expansion**: The character expansion factor.

**RETURNS**  
None

**DESCRIPTION**  
This function creates an output primitive attribute which sets the character expansion factor attribute. Only the magnitude of the specified expansion is considered. The specified character expansion factor is compared to the minimum and maximum character expansion factors. These values depend on the font files that are in the font groups in the selected font table entry, which in turn depend on the font files opened. For example, if all scalable and rotatable stroke fonts are open, then a continuous number of expansions are supported. If the expansion is smaller than the minimum character expansion factor, the minimum value is used. If the expansion is larger than the maximum character expansion factor, the maximum value is used.

**ERRORS**  
- **BadPEXRenderer**: The specified renderer resource identifier is invalid.
- **BadPEXStructure**: The specified structure resource identifier is invalid.

**SEE ALSO**  
PEXSetIndividualASF(3)
### NAME
PEXSetCharHeight - Set Character Height

### SYNTAX
```c
void PEXSetCharHeight(Display *display, XID resource_id, PEXOCRequestType req_type,
                      double height)
```

### PARAMETERS
- **display**: A pointer to a display structure returned by a successful `XOpenDisplay` call.
- **resource_id**: The resource identifier of the renderer or structure.
- **req_type**: The request type for the output command ( `PEXOCRender`, `PEXOCStore`, `PEXOCRenderSingle` or `PEXOCStoreSingle`).
- **height**: The text character height.

### RETURNS
None

### DESCRIPTION
This function creates an output primitive attribute which sets the character height value attribute. If the specified `height` or the computed width is not supported, the height or width is mapped to the nearest supported character height or width. These values depend on the font files that are in the font groups in the selected font table entry, which in turn depend on which X or PEX fonts are open. For example, if the client opens all scalable and rotatable stroke fonts, a continuous number of character sizes are supported. The `height` is specified in the text local coordinate system.

### ERRORS
- **BadPEXRenderer**: The specified renderer resource identifier is invalid.
- **BadPEXStructure**: The specified structure resource identifier is invalid.
NAME
PEXSetCharSpacing - Set Character Spacing

SYNTAX
void PEXSetCharSpacing(Display *display, XID resource_id, PEXOCRequestType
req_type, double spacing)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay
call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOC-
Store, PEXOCRenderSingle or PEXOCStoreSingle).
spacing The text character spacing factor.

RETURNS None

DESCRIPTION This function creates an output primitive attribute which sets the character
spacing attribute. The character spacing attribute is expressed as a fraction of the font’s nominal char-
acter height.

ERRORS
BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO PEXSetIndividualASF(3)
NAME        PEXSetCharUpVector - Set Character Up Vector

SYNTAX      void PEXSetCharUpVector(Display *display, XID resource_id, PEXOCRequestType
           req_type, PEXVector2D *vector)

PARAMETERS  display       A pointer to a display structure returned by a successful XOpenDisplay
            call.

            resource_id  The resource identifier of the renderer or structure.

            req_type     The request type for the output command (PEXOCRender, PEXOC-
            Store, PEXOCRenderSingle or PEXOCStoreSingle).

            vector       The text character up vector.

RETURNS     None

DESCRIPTION This function creates an output primitive attribute which sets the text up vector attribute.
The vector is specified in the text local coordinate system defined by the direction vectors
associated with the annotation text primitive. If the annotation text up vector is degenerate (it has a length of zero), a value <0,1> is used.

DATA STRUCTURES See PEXlib.h.

ERRORS      BadPEXRenderer
            The specified renderer resource identifier is invalid.

            BadPEXStructure
            The specified structure resource identifier is invalid.
NAME       PEXSetColorApproxIndex - Set Color Approximation Index

SYNTAX    void PEXSetColorApproxIndex(Display *display, XID resource_id, PEXOCRequestType req_type, unsigned int index)

PARAMETERS  
  display       A pointer to a display structure returned by a successful XOpenDisplay call.
  resource_id   The resource identifier of the renderer or structure.
  req_type      The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
  index         The color approximation table index.

RETURNS      None

DESCRIPTION  This function creates an output primitive attribute which sets the color approximation index attribute. If the specified index is not defined, index zero is used. If index zero is not defined, the result is implementation-dependent.

ERRORS       
  BadPEXOutputCommand          The color approximation index value exceeds 65534.
  BadPEXRenderer               The specified renderer resource identifier is invalid.
  BadPEXStructure              The specified structure resource identifier is invalid.

SEE ALSO    PEXCreateLookupTable(3)  
             PEXGetImpDepConstants(3)

modified November 1995
NAME
PEXSetCurveApprox - Set Curve Approximation Method

SYNTAX
void PEXSetCurveApprox(Display *display, XID resource_id, PEXOCRequestType
req_type, int method, double tolerance)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay
call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOC-
Store, PEXOCRenderSingle or PEXOCStoreSingle).
method The curve approximation method (PEXApproxImpDep, PEXApprox-
ConstantBetweenKnots, PEXApproxWCChordalSize, PEXApprox-
NPCChordalSize, PEXApproxDCChordalSize, PEXCurveAp-
proxWCCChordalDev, PEXCurveApproxNPCChordalDev, PEXCur-
veApproxDCChordalDev, PEXApproxWCRelative, PEXApproxNP-
CRelative, PEXApproxDCRelative).
tolerance The curve approximation tolerance (specific to each method).

RETURNS
None

DESCRIPTION
This function creates an output primitive attribute which sets the curve approximation
attribute. If the specified method is not supported, an implementation-dependent method
(method 1) is used. Supported values for curve approximation are inquirable via PEX-
GetEnumTypeInfo(3). The tolerance value is provided to indicate the desired accuracy of
the approximation, and is used in different ways for the different methods.

ERRORS
BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO
PEXSetIndividualASF(3)
PEXGetEnumTypeInfo(3)

modified November 1995
**NAME**
PEXSetDepthCueIndex - Set Depth Cue Index

**SYNTAX**
```c
void PEXSetDepthCueIndex(Display *display, XID resource_id, PEXOCRequestType req_type, unsigned int index)
```

**PARAMETERS**
- **display**: A pointer to a display structure returned by a successful XOpenDisplay call.
- **resource_id**: The resource identifier of the renderer or structure.
- **req_type**: The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- **index**: The depth cue table index.

**RETURNS**
None

**DESCRIPTION**
This function creates an output primitive attribute which sets the depth-cue index attribute. If the specified index is not defined, index zero is used. If index zero is not defined, depth cueing is turned off.

**ERRORS**
- **BadPEXOutputCommand**: The depth cue index value exceeds 65534.
- **BadPEXRenderer**: The specified renderer resource identifier is invalid.
- **BadPEXStructure**: The specified structure resource identifier is invalid.

**SEE ALSO**
PEXCreateLookupTable(3)
NAME
PEXSetEchoColor - PEX Escape to set Echo Color for Specified Renderer

SYNTAX
void PEXSetEchoColor(Display *display, PEXRenderer renderer, int color_type, PEXColor *color)

PARAMETERS
display  A pointer to a display structure returned by a successful XOpenDisplay call.
renderer  The resource identifier of a renderer.
color_type  The type of color (PEXColorTypeIndexed, PEXColorTypeRGB, PEXColorTypeCIE, PEXColorTypeHSV, PEXColorTypeHLS, PEXColorTypeRGB8, PEXColorTypeRGB16).
color  The echo color.

RETURNS
None

DESCRIPTION
This function is a convenient way to access the PEXEscapeSetEchoColor escape to set the echo color for use by a renderer when echoing primitives.
A renderer’s echo color can be changed at any time during rendering and is not inquirable through the renderer’s modification dynamics (see PEXGetRendererDynamics(3)). Lighting and shading for echoed primitives is implementation-dependent. A renderer’s echo color cannot be inquired and the default value is implementation-dependent. Support of this escape is inquirable via PEXGetEnumTypeInfo(3).

DATA STRUCTURES
typedef XID  PEXRenderer;
typedef union {
    PEXColorIndexed  indexed;
    PEXColorRGB  rgb;
    PEXColorHSV  hsv;
    PEXColorHLS  hls;
    PEXColorCIE  cie;
    PEXColorRGB8  rgb8;
    PEXColorRGB16  rgb16;
} PEXColor;
typedef struct {
    PEXTableIndex  index;
    unsigned short reserved;
} PEXColorIndexed;
typedef struct {
    float  red;
    float  green;
    float  blue;
}
typedef struct {
    float hue;
    float saturation;
    float value;
} PEXColorHSV;

typedef struct {
    float hue;
    float lightness;
    float saturation;
} PEXColorHLS;

typedef struct {
    float x;
    float y;
    float z;
} PEXColorCIE;

typedef struct {
    unsigned char red;
    unsigned char green;
    unsigned char blue;
    unsigned char reserved;
} PEXColorRGB8;

typedef struct {
    unsigned short red;
    unsigned short green;
    unsigned short blue;
    unsigned short reserved;
} PEXColorRGB16;

**ERRORS**

**BadPEXColorType**
The specified color type is invalid or unsupported.

**BadPEXRenderer**
The specified renderer resource identifier is invalid.

**BadValue**
The `PEXEscapeSetEchoColor` escape is unsupported.

**SEE ALSO**
PEXEscape(3)
NAME  
PEXSetEdgeBundleIndex - Set Edge Bundle Index

SYNTAX  
void PEXSetEdgeBundleIndex(Display ∗display, XID resource_id, PEXOCRequestType req_type, unsigned int index)

PARAMETERS  
display  
A pointer to a display structure returned by a successful XOpenDisplay call.

resource_id  
The resource identifier of the renderer or structure.

req_type  
The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).

index  
The edge bundle table index.

RETURNS  
None

DESCRIPTION  
This function creates an output primitive attribute which sets the edge bundle index attribute. If an undefined index is specified, default bundle index one is used. If index one is not defined, the edge bundle attributes are set as follows: surface edges are turned off, the edge type is solid, the edge width is 1.0 and the color is indexed color one. An edge bundle table index of 0 will produce a BadPEXOutputCommand error.

ERRORS  
BadPEXOutputCommand  
The output command contains an invalid value.

BadPEXRender  
The specified renderer resource identifier is invalid.

BadPEXStructure  
The specified structure resource identifier is invalid.

SEE ALSO  
PEXSetIndividualASF(3)  
PEXCreateLookupTable(3)
NAME  
PEXSetEditingMode - Set Structure Editing Mode

SYNTAX  
void PEXSetEditingMode(Display *display, PEXStructure structure, int mode)

PARAMETERS  
display  
A pointer to a display structure returned by a successful XOpenDisplay call.

structure  
The resource identifier of the structure.

mode  
The editing mode (PEXStructureInsert or PEXStructureReplace).

RETURNS  
None

DESCRIPTION  
This function sets the editing mode for the structure specified. The editing mode specifies how editing operations affect the structure. If the editing mode is PEXStructureInsert, subsequent requests to create structure elements cause elements to be inserted into the structure. The element pointer is then incremented by the number of elements inserted. If the editing mode is PEXStructureReplace, output requests that create structure elements cause structure elements to replace elements, starting at the location specified by the element pointer.

DATA STRUCTURES  
typedef XID PEXStructure;

ERRORS  
BadPEXStructure
The specified structure resource identifier is invalid.

BadValue
The specified value for mode is invalid.

SEE ALSO  
PEXCreateStructure(3)
<table>
<thead>
<tr>
<th><strong>NAME</strong></th>
<th>PEXSetElementPtr - Set Structure Element Pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SYNTAX</strong></td>
<td>void PEXSetElementPtr(Display *display, PEXStructure structure, int whence, long offset)</td>
</tr>
<tr>
<td><strong>PARAMETERS</strong></td>
<td></td>
</tr>
<tr>
<td>display</td>
<td>A pointer to a display structure returned by a successful XOpenDisplay call.</td>
</tr>
<tr>
<td>structure</td>
<td>The resource identifier of the structure.</td>
</tr>
<tr>
<td>whence</td>
<td>A value specifying with offset, the element pointer position (PEXBeginning, PEXCurrent, PEXEnd).</td>
</tr>
<tr>
<td>offset</td>
<td>The offset from whence which specifies the element pointer position.</td>
</tr>
<tr>
<td><strong>RETURNS</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>DESCRIPTION</strong></td>
<td>This function sets the element pointer for the structure specified to the position specified. If either computed offset is less than zero, it is set to zero before obtaining the element information. If either computed offset is greater than the number of elements in the structure, it is set to the offset of the last structure element in the structure.</td>
</tr>
<tr>
<td><strong>DATA STRUCTURES</strong></td>
<td>typedef XID PEXStructure;</td>
</tr>
<tr>
<td><strong>ERRORS</strong></td>
<td>BadPEXStructure</td>
</tr>
<tr>
<td></td>
<td>The specified structure resource identifier is invalid.</td>
</tr>
<tr>
<td>BadValue</td>
<td>The specified value for whence parameter is invalid.</td>
</tr>
<tr>
<td><strong>SEE ALSO</strong></td>
<td>PEXCreateStructure(3)</td>
</tr>
</tbody>
</table>
NAME       PEXSetElementPtrAtLabel - Set Structure Element Pointer at Label

SYNTAX     void PEXSetElementPtrAtLabel(Display *display, PEXStructure structure, long label, long offset)

PARAMETERS display     A pointer to a display structure returned by a successful XOpenDisplay call.
structure    The resource identifier of the structure.
label       The value of the label.
offset       The offset from the label.

RETURNS None

DESCRIPTION This function sets the element pointer for the specified structure at a position denoted by the label. A search is conducted for the next occurrence of the label, starting at the current element pointer position plus one and proceeding in the forward direction. If label is found, the element pointer for the structure is set to the location of the label plus the value of the specified offset. If label is not found, the structure's element pointer is left unchanged.

DATA STRUCTURES
typedef XID PEXStructure;  

ERRORS
BadPEXLabel
The specified label does not exist.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO PEXCreateStructure(3)  
PEXLabel(3)
NAME
PEXSetFacetCullingMode - Set Facet Culling Mode

SYNTAX
void PEXSetFacetCullingMode(Display *display, XID resource_id, PEXOCRequestType req_type, int mode)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
mode The facet culling mode (PEXNone, PEXBackFaces, PEXFrontFaces).

RETURNS None

DESCRIPTION This function creates an output primitive attribute which sets the facet culling mode attribute.

ERRORS BadPEXOutputCommand
The output command contains an invalid value.

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXSetFacetDistinguishFlag - Set Facet Distinguish Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td>void PEXSetFacetDistinguishFlag(Display *display, XID resource_id, PEXOCRequestType req_type, int flag)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>display: A pointer to a display structure returned by a successful XOpenDisplay call.</td>
</tr>
<tr>
<td></td>
<td>resource_id: The resource identifier of the renderer or structure.</td>
</tr>
<tr>
<td></td>
<td>req_type: The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).</td>
</tr>
<tr>
<td></td>
<td>flag: The facet distinguish flag (True or False).</td>
</tr>
<tr>
<td>RETURNS</td>
<td>None</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>This function creates an output primitive attribute which sets the facet distinguish flag attribute. Values for the facet distinguish flag are True (use back-face attributes to renderer surfaces) or False (use front-face attributes to renderer surfaces).</td>
</tr>
<tr>
<td>ERRORS</td>
<td>BadPEXOutputCommand: The output command contains an invalid value.</td>
</tr>
<tr>
<td></td>
<td>BadPEXRenderer: The specified renderer resource identifier is invalid.</td>
</tr>
<tr>
<td></td>
<td>BadPEXStructure: The specified structure resource identifier is invalid.</td>
</tr>
</tbody>
</table>
NAME       PEXSetGlobalTransform - Set Global Transformation 3D

SYNTAX     void PEXSetGlobalTransform(Display *display, XID resource_id, PEXOCRequestType req_type, PEXMatrix transform)

PARAMETERS display A pointer to a display structure returned by a successful XGetDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
transform The new global transformation matrix.

RETURNS None

DESCRIPTION This function creates an output primitive attribute which replaces the current global transformation matrix with the specified matrix. The matrix is stored in row-major order.

DATA STRUCTURES See PEXlib.h.

ERRORS BadPEXRenderer The specified renderer resource identifier is invalid.
BadPEXStructure The specified structure resource identifier is invalid.

SEE ALSO PEXSetGlobalTransform2D(3)
NAME  PEXSetGlobalTransform2D - Set Global Transformation 2D

SYNTAX  void PEXSetGlobalTransform2D(Display *display, XID resource_id, PEXOCRequestType req_type, PEXMatrix3x3 transform)

PARAMETERS  display  A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id  The resource identi®er of the renderer or structure.
req_type  The request type for the output command ( PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
transform  The new global transformation matrix.

RETURNS  None

DESCRIPTION  This function creates an output primitive attribute which replaces the current global transformation matrix with the speci®ed matrix. This output command is similar to PEXSetGlobalTransform(3) except that the global transformation matrix is specified as a $3 \times 3$ matrix. Before replacement of the global transformation matrix, the $3 \times 3$ matrix represented by

\[
\begin{bmatrix}
a & b & c \\
d & e & f \\
g & h & j
\end{bmatrix}
\]

is expanded to a $4 \times 4$ matrix as follows:

\[
\begin{bmatrix}
a & b & c \\
d & e & f \\
g & h & j
\end{bmatrix} \rightarrow \begin{bmatrix}
a & b & 0 & c \\
d & e & 0 & f \\
0 & 0 & 1 & 0 \\
g & h & 0 & j
\end{bmatrix}
\]

DATA STRUCTURES  See PEXlib.h.

ERRORS  BadPEXRenderer
  The specified renderer resource identi®er is invalid.
BadPEXStructure
  The specified structure resource identi®er is invalid.

SEE ALSO  PEXSetGlobalTransform(3)
**NAME**
PEXSetHLHSRID - Set Hidden-Line Hidden-Surface Removal Identifier

**SYNTAX**
```c
void PEXSetHLHSRID(Display *display, XID resource_id, PEXOCRequestType req_type,
                    unsigned long hlhsr_id)
```

**PARAMETERS**
- `display` A pointer to a display structure returned by a successful `XOpenDisplay` call.
- `resource_id` The resource identifier of the renderer or structure.
- `req_type` The request type for the output command ( `PEXOCRender`, `PEXOCStore`, `PEXOCRenderSingle` or `PEXOCStoreSingle`).
- `hlhsr_id` The HLHSR identifier ( `PEXHLHSRIDDisable` or `PEXHLHSRIDEnable`).

**RETURNS**
None

**DESCRIPTION**
This function creates an output primitive attribute which sets the HLHSR identifier attribute. This output command is a no-op for all standard HLHSR modes except `PEXHLHSRZBufferID`. If the renderer’s HLHSR mode is set to `PEXHLHSRZBufferID`, then a HLHSR identifier of `PEXHLHSRIDEnable` will disable z-buffering and a HLHSR identifier of `PEXHLHSRIDEnable` will enable z-buffering. For non-standard HLHSR modes, the effect of this output command is implementation-dependent.

**ERRORS**
- **BadPEXRenderer**
  The specified renderer resource identifier is invalid.
- **BadPEXStructure**
  The specified structure resource identifier is invalid.
NAME
PEXSetIndividualASF - Set Individual Aspect Source Flags

SYNTAX
void PEXSetIndividualASF(Display *display, XID resource_id, PEXOCRequestType
req_type, unsigned long attribute, int asf)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay
call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOC-
Store, PEXOCRenderSingle or PEXOCStoreSingle).
attribute The ASF attribute name (see the Description).
asf The attribute source flag value (PEXIndividual or PEXBundled).

RETURNS
None

DESCRIPTION
This function creates an output primitive attribute which sets the aspect source flag for a
single, individual attribute. For the individual ASF attribute name, valid values are:

PEXASFBFInteriorStyle PEXASFBFInteriorStyleIndex
PEXASFBFReflectionAttr PEXASFBFReflectionModel
PEXASFBFSurfaceColor PEXASFBFSurfaceInterp
PEXASFCurveApprox PEXASFCharSpacing
PEXASFCurveApprox PEXASFInteriorStyle
PEXASFInteriorStyleIndex PEXASFLineColor
PEXASFLineType PEXASFLINEWidth
PEXASFMarkerColor PEXASFMarkerScale
PEXASFMarkerType PEXASFPolylineInterp
PEXASFReflectionAttr PEXASFReflectionModel
PEXASFSurfaceApprox PEXASFSurfaceColor
PEXASFSurfaceEdgeColor PEXASFSurfaceEdgeType
PEXASFSurfaceEdgeWidth PEXASFSurfaceEdges
PEXASFSurfaceInterp PEXASFTColor
PEXASFTextFontIndex PEXASFTextPrec

ERRORS
BadPEXOutputCommand
The output command contains an invalid value.

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.
**NAME**  
PEXSetInteriorBundleIndex - Set Interior Bundle Index

**SYNTAX**  
void PEXSetInteriorBundleIndex(Display *display, XID resource_id, PEXOCRequestType req_type, unsigned int index)

**PARAMETERS**  
- **display**  
  A pointer to a display structure returned by a successful XOpenDisplay call.

- **resource_id**  
  The resource identifier of the renderer or structure.

- **req_type**  
  The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).

- **index**  
  The interior bundle table index.

**RETURNS**  
None

**DESCRIPTION**  
This function creates an output primitive attribute which sets the interior bundle index attribute. If the specified index is not defined, the default index one is used. If index one is not defined, the following bundle values are used: interior style is PEXInteriorStyleHollow; interior style index is one; surface color is indexed color one; reflection attributes are ambient, diffuse and specular coefficient of 1.0, specular concentration and transmission coefficient of 0.0 and specular color of indexed color one; reflection model is PEXReflectionNone; surface interpolation is PEXSurfaceInterpNone; and surface approximation is method 1 (PEXApproxImpDep) with u and v tolerances of 1.0. The back-facing attributes are set to identical values as the front-facing attributes.

An interior bundle table index of 0 will produce a BadPEXOutputCommand error.

**ERRORS**  
- **BadPEXOutputCommand**  
  The output command contains an invalid value.

- **BadPEXRenderer**  
  The specified renderer resource identifier is invalid.

- **BadPEXStructure**  
  The specified structure resource identifier is invalid.

**SEE ALSO**  
PEXSetIndividualASF(3)  
PEXCreateLookupTable(3)

modified November 1995
**NAME**
PEXSetInteriorStyle - Set Surface Interior Style

**SYNTAX**
void PEXSetInteriorStyle(Display *display, XID resource_id, PEXOCRequestType req_type, int style)

**PARAMETERS**
display  A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id  The resource identifier of the renderer or structure.
req_type  The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
style  The interior style (PEXInteriorStyleHollow, PEXInteriorStyleSolid, PEXInteriorStylePattern, PEXInteriorStyleHatch, PEXInteriorStyleEmpty).

**RETURNS**
None

**DESCRIPTION**
This function creates an output primitive attribute which sets the surface interior style attribute. If the specified style is not supported, PEXInteriorStyleHollow is used. Supported values for surface interior style are inquirable via PEXGetEnumTypeInfo(3).

**ERRORS**
BADPEXOUTPUTCOMMAND  The output command contains an invalid value.
BADPEXRENDERER  The specified renderer resource identifier is invalid.
BADPEXSTRUCTURE  The specified structure resource identifier is invalid.

**SEE ALSO**
PEXGetEnumTypeInfo(3)
PEXSetIndividualASF(3)

modified November 1995
NAME

PEXSetInteriorStyleIndex - Set Surface Interior Style Index

SYNTAX

```c
void PEXSetInteriorStyleIndex(Display *display, XID resource_id, PEXOCRequestType req_type, int index);
```

PARAMETERS

- `display` A pointer to a display structure returned by a successful `XOpenDisplay` call.
- `resource_id` The resource identifier of the renderer or structure.
- `req_type` The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- `index` The interior style index.

RETURNS

None

DESCRIPTION

This function creates an output primitive attribute which sets the interior style `index` attribute. If the current interior style is PEXInteriorStylePattern or PEXInteriorStyleHatch, the specified `index` is used to further define the rendering style of front-facing surface primitives. For PEXInteriorStylePattern, if the specified pattern table index is not defined, table index one is used. For PEXInteriorStyleHatch, the index determines the hatch style. If the specified hatch style is not supported, style one is used. If style one is not supported, the result is implementation-dependent. Supported values for hatch style are inquiryable via PEXGetEnumTypeInfo(3).

ERRORS

- BadPEXRenderer
  - The specified renderer resource identifier is invalid.
- BadPEXStructure
  - The specified structure resource identifier is invalid.

SEE ALSO

PEXSetIndividualASF(3)
PEXCreateLookupTable(3)
PEXGetEnumTypeInfo(3)
### NAME
PEXSetLightSourceState - Set Light Source State

### SYNTAX
```c
void PEXSetLightSourceState(Display *display, XID resource_id, PEXOCRequestType req_type, unsigned int enable_count, PEXTableIndex *enable, unsigned int disable_count, PEXTableIndex *disable)
```

### PARAMETERS
- **display**: A pointer to a display structure returned by a successful `XOpenDisplay` call.
- **resource_id**: The resource identifier of the renderer or structure.
- **req_type**: The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- **enable_count**: The number of lights to enable.
- **enable**: An array of lights to enable.
- **disable_count**: The number of lights to disable.
- **disable**: An array of lights to disable.

### RETURNS
None

### DESCRIPTION
This function creates an output primitive attribute which sets the light source state attribute. Each element in the `enable` list activates the light represented by the corresponding light table entry and each element in the `disable` list deactivates the light represented by the corresponding light table entry. A light listed in both the `enable` list and the `disable` list, or a light index of 0 will produce a `BadPEXOutputCommand` error. Otherwise, if any light in the `enable` or `disable` list references an undefined light table entry, the light is ignored.

### DATA STRUCTURES
```c
typedef unsigned short PEXTableIndex;
```

### ERRORS
- **BadPEXOutputCommand**: The output command contains an invalid value.
- **BadPEXRenderer**: The specified renderer resource identifier is invalid.
- **BadPEXStructure**: The specified structure resource identifier is invalid.

### SEE ALSO
PEXCreateLookupTable(3)

---

modified November 1995
NAME

PEXSetLineBundleIndex - Set Line Bundle Index

SYNTAX

void PEXSetLineBundleIndex(Display *display, XID resource_id, PEXOCRequestType req_type, unsigned int index)

PARAMETERS

display
The resource identifier of the renderer or structure.
resource_id
The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
index
The line bundle table index.

RETURNS

None

DESCRIPTION

This function creates an output primitive attribute which sets the line bundle index attribute. If an undefined line bundle index is specified, the default bundle index one is used. If index one is not defined, the following values are used: line type is PEXLineTypeSolid; polyline interpolation is PEXPolylineInterpNone; curve approximation is method 1 (PEXApproxImpDep) with a tolerance of 1.0; line width is 1.0 and line color is index color one.

A line bundle table index of 0 will produce a BadPEXOutputCommand error.

ERRORS

BadPEXOutputCommand
The output command contains an invalid value.
BadPEXRenderer
The specified renderer resource identifier is invalid.
BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO

PEXSetIndividualASF(3)
PEXCreateLookupTable(3)

modified November 1995
**NAME**
PEXSetLineColor - Set Line Color

**SYNTAX**
```c
void PEXSetLineColor(Display *display, XID resource_id, PEXOCRequestType req_type,
int color_type, PEXColor *color)
```

**PARAMETERS**
- `display`  A pointer to a display structure returned by a successful `XOpenDisplay` call.
- `resource_id`  The resource identifier of the renderer or structure.
- `req_type`  The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- `color_type`  The type of color (PEXColorTypeIndexed, PEXColorTypeRGB, PEXColorTypeCIE, PEXColorTypeHSV, PEXColorTypeHLS, PEXColorTypeRGB8, PEXColorTypeRGB16).
- `color`  A pointer to the line color.

**RETURNS**
None

**DESCRIPTION**
This function creates an output primitive attribute which sets the line color attribute. The attribute is set to either an indexed color or a direct color value, depending on the color type. If the color type is PEXColorTypeIndexed and the specified color index is not defined, color index one is used. If color index one is not defined, the result is implementation-dependent.

**DATA STRUCTURES**
See PEXlib.h.

**ERRORS**
- BadPEXOutputCommand  The output command contains an invalid value, e.g. the color type is PEXColorTypeIndexed, and the color index value exceeds 65534.
- BadPEXRenderer  The specified renderer resource identifier is invalid.
- BadPEXStructure  The specified structure resource identifier is invalid.

**SEE ALSO**
PSEXetIndividualASF(3)
PSEXCreateLookupTable(3)

---

modified November 1995
## NAME
PEXSetLineColorIndex - Set Line Color Index

## SYNTAX
```c
void PEXSetLineColorIndex(Display *display, XID resource_id, PEXOCRequestType req_type, unsigned int index)
```

## PARAMETERS
- **display**: A pointer to a display structure returned by a successful `XOpenDisplay` call.
- **resource_id**: The resource identifier of the renderer or structure.
- **req_type**: The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- **index**: The color table index for lines.

## RETURNS
None

## DESCRIPTION
This function creates an output primitive attribute which sets the line color attribute to an indexed color with the value indicated by `index`. If the color `index` is not defined, color index one is used. If color index one is not defined, the result is implementation-dependent.

## ERRORS
- **BadPEXOutputCommand**: The color index value exceeds 65534.
- **BadPExRenderer**: The specified renderer resource identifier is invalid.
- **BadPEXStructure**: The specified structure resource identifier is invalid.

## SEE ALSO
- PEXSetIndividualASF(3)
- PEXCreateLookupTable(3)
NAME  PEXSetLineType - Set Line Type

SYNTAX  void PEXSetLineType(Display ∗display, XID resource_id, PEXOCRequestType req_type,
                         int line_type)

PARAMETERS  
  display  A pointer to a display structure returned by a successful XOpenDisplay
            call.
  resource_id  The resource identifier of the renderer or structure.
  req_type  The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or
            PEXOCStoreSingle).
  line_type  The line type (PEXLineTypeSolid, PEXLineTypeDashed, PEXLineTypeDotted, PEXLineTypeDashDot).

RETURNS  None

DESCRIPTION  This function creates an output primitive attribute which sets the line type attribute. If
the specified line type is not supported, PEXLineTypeSolid is used. Supported values
for line type are inquirable via PEXGetEnumTypeInfo.

ERRORS  BadPEXRenderer
            The specified renderer resource identifier is invalid.
BadPEXStructure
            The specified structure resource identifier is invalid.

SEE ALSO  PEXGetEnumTypeInfo
          PEXSetIndividualASF
NAME
PEXSetLineWidth - Set Line Width

SYNTAX
void PEXSetLineWidth(Display *display, XID resource_id, PEXOCRequestType req_type,
                        double width)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
width The line width scale factor.

RETURNS
None

DESCRIPTION
This function creates an output primitive attribute which sets the line width attribute. The specified width is used as a scale factor. This scale factor is used to increase or decrease the width (in pixels) from the nominal line width for the display device. The result is mapped to the nearest supported line width.

ERRORS
BadPEXRenderer
The specified renderer resource identifier is invalid.
BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO
PEXGetImpDepConstants(3)
PEXSetIndividualASF(3)
### NAME
PEXSetLocalTransform - Set Local Transformation 3D

### SYNTAX
```c
void PEXSetLocalTransform(Display *display, XID resource_id, PEXOCRequestType req_type, int composition, PEXMatrix transform)
```

### PARAMETERS
- **display**: A pointer to a display structure returned by a successful `XOpenDisplay` call.
- **resource_id**: The resource identifier of the renderer or structure.
- **req_type**: The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- **composition**: The composition rule for combining the new matrix with the current local transform (PEXPreConcatenate, PEXPostConcatenate, PEXReplace).
- **transform**: The new local transformation matrix.

### RETURNS
None

### DESCRIPTION
This function creates an output primitive attribute which modifies the local transformation matrix. If the composition type is PEXPreConcatenate, the specified matrix is pre-concatenated to the current local transformation matrix \((L' = L \times T)\). If the composition type is PEXPostConcatenate, the specified matrix is post-concatenated to the current local transformation matrix \((L' = T \times L)\). If the composition type is PEXReplace, the specified matrix replaces the current local transformation matrix \((L' = T)\). The composite matrix is then recomputed using the current global transformation and the new local transformation matrix \((C = G \times L')\).

### DATA STRUCTURES
See PEXlib.h.

### ERRORS
- **BadPEXOutputCommand**: The output command contains an invalid value.
- **BadPEXRenderer**: The specified renderer resource identifier is invalid.
- **BadPEXStructure**: The specified structure resource identifier is invalid.

### SEE ALSO
PEXSetLocalTransform2D(3)

---

modified November 1995
NAME
PEXSetLocalTransform2D - Set Local Transformation 2D

SYNTAX
void PEXSetLocalTransform2D(Display *display, XID resource_id, PEXOCRequestType req_type, int composition, PEXMatrix3x3 transform)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
composition The composition rule for combining the new matrix with the current local transform (PEXPreConcatenate, PEXPostConcatenate, PEXReplace).
transform The new local transformation matrix.

RETURNS None

DESCRIPTION
This function creates an output primitive attribute which modifies the local transformation matrix. This output command is similar to PEXSetLocalTransform(3) except that the local transformation matrix is specified as a 3x3 matrix. Before modification of the local transformation matrix, the 3x3 matrix represented by

\[
\begin{bmatrix}
a & b & c \\
d & e & f \\
g & h & j \\
\end{bmatrix}
\]

is expanded to a 4x4 matrix as follows:

\[
\begin{bmatrix}
a & b & c & 0 \\
d & e & f & 0 \\
g & h & j & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

DATA STRUCTURES
See PEXlib.h.

ERRORS
BadPEXOutputCommand
The output command contains an invalid value.

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.
SEE ALSO

PEXSetLocalTransform(3)
**NAME**

PEXSetMarkerBundleIndex - Set Marker Bundle Index

**SYNTAX**

```c
void PEXSetMarkerBundleIndex(Display *display, XID resource_id, PEXOCRequestType req_type, unsigned int index)
```

**PARAMETERS**

- `display`: A pointer to a display structure returned by a successful `XOpenDisplay` call.
- `resource_id`: The resource identifier of the renderer or structure.
- `req_type`: The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- `index`: The marker bundle table index.

**RETURNS**

None

**DESCRIPTION**

This function creates an output primitive attribute which sets the marker bundle index attribute. If an undefined index is specified, the default bundle index one is used. If index one is undefined, the default values are used: marker type is PEXMarkerAsterisk; marker scale is 1.0 and marker color is indexed color one.

A marker bundle table index of 0 will produce a BadPEXOutputCommand error.

**ERRORS**

- **BadPEXOutputCommand**: The output command contains an invalid value.
- **BadPEXRenderer**: The specified renderer resource identifier is invalid.
- **BadPEXStructure**: The specified structure resource identifier is invalid.

**SEE ALSO**

PEXCreateLookupTable(3)

PEXSetIndividualASF(3)
NAME
PEXSetMarkerColor - Set Marker Color

SYNTAX
void PEXSetMarkerColor(Display *display, XID resource_id, PEXOCRequestType req_type, int color_type, PEXColor *color)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
color_type The type of color (PEXColorTypeIndexed, PEXColorTypeRGB, PEXColorTypeCIE, PEXColorTypeHSV, PEXColorTypeHLS, PEXColorTypeRGB8, PEXColorTypeRGB16).
color A pointer to the marker color.

RETURNS
None

DESCRIPTION
This function creates an output primitive attribute which sets the marker color attribute. The attribute is set to either an indexed color or a direct color value, depending on the color type. If the color type is PEXColorTypeIndexed and the specified color index is not defined, color index one is used. If color index one is not defined, the result is implementation-dependent.

DATA STRUCTURES
See PEXlib.h.

ERRORS
BadPEXOutputCommand
The output command contains an invalid value, e.g. the color type is PEXColorTypeIndexed, and the color index value exceeds 65534.

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO
PEXSetIndividualASF(3)
PEXCreateLookupTable(3)

modified November 1995
NAME
PEXSetMarkerColorIndex - Set Marker Color Index

SYNTAX
void PEXSetMarkerColorIndex(Display *display, XID resource_id, PEXOCRequestType
req_type, unsigned int index)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay
       call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOC-
       Store, PEXOCRenderSingle or PEXOCStoreSingle).
index The color table index for markers.

RETURNS
None

DESCRIPTION
This function creates an output primitive attribute which sets the marker color attribute
to an indexed color with the value indicated by index. If the color index is not defined,
color index one is used. If color index one is not defined, the result is implementation-
dependent.

ERRORS
BadPEXOutputCommand
   The color index value exceeds 65534.
BadPEXRenderer
   The specified renderer resource identifier is invalid.
BadPEXStructure
   The specified structure resource identifier is invalid.

SEE ALSO
PEXSetIndividualASF(3)
PEXCreateLookupTable(3)

modified November 1995
PEXSetMarkerScale - Set Marker Scale

void PEXSetMarkerScale(Display *display, XID resource_id, PEXOCRequestType req_type, double scale)

A pointer to a display structure returned by a successful XOpenDisplay call.

The resource identifier of the renderer or structure.

The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).

The marker scale factor.

None

This function creates an output primitive attribute which sets the marker scale attribute. Scale is multiplied by the nominal marker size for the display device to produce a marker size mapped to the nearest supported marker size.

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

PEXGetImpDepConstants(3)
PEXSetIndividualASF(3)

modified November 1995
**NAME**  
PEXSetMarkerType - Set Marker Type

**SYNTAX**  
```c
void PEXSetMarkerType(Display ∗display, XID resource_id, PEXOCRequestType req_type, int marker_type)
```

**PARAMETERS**  
- `display`  
  A pointer to a display structure returned by a successful `XOpenDisplay` call.
- `resource_id`  
  The resource identifier of the renderer or structure.
- `req_type`  
  The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- `marker_type`  
  The marker type (PEXMarkerDot, PEXMarkerCross, PEXMarkerAsterisk, PEXMarkerCircle, PEXMarkerX).

**RETURNS**  
None

**DESCRIPTION**  
This function creates an output primitive attribute which sets the marker type attribute. If the specified marker type is not supported, PEXMarkerAsterisk is used. Supported values for marker type are inquirable via `PEXGetEnumTypeInfo(3)`.

**ERRORS**  
- BadPEXRenderer  
  The specified renderer resource identifier is invalid.
- BadPEXStructure  
  The specified structure resource identifier is invalid.

**SEE ALSO**  
PEXGetEnumTypeInfo(3)  
PEXSetIndividualASF(3)
### NAME
PEXSetModelClipFlag - Set Model Clipping Flag

### SYNTAX
```c
void PEXSetModelClipFlag(Display *display, XID resource_id, PEXOCRequestType req_type, int flag)
```

### PARAMETERS
- **display**
  A pointer to a display structure returned by a successful `XOpenDisplay` call.
- **resource_id**
  The resource identifier of the renderer or structure.
- **req_type**
  The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- **flag**
  The model clipping flag (PEXClip or PEXNoClip).

### RETURNS
None

### DESCRIPTION
This function creates an output primitive attribute which sets the model clip attribute. Values for the model clip flag are PEXClip (enable modeling clipping) and PEXNoClip (disable modeling clipping).

### ERRORS
- **BadPEXOutputCommand**
  The output command contains an invalid value.
- **BadPEXRenderer**
  The specified renderer resource identifier is invalid.
- **BadPEXStructure**
  The specified structure resource identifier is invalid.

### SEE ALSO
- PEXSetModelClipVolume(3)
- PEXSetModelClipVolume2D(3)
- PEXRestoreModelClipVolume(3)
NAME
PEXSetModelClipVolume - Set Model Clip Volume 3D

SYNTAX
void PEXSetModelClipVolume(Display ∗display, XID resource_id, PEXOCRequestType
req_type, int op, unsigned int count, PEXHalfSpace ∗half_spaces)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay
call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOC-
Store, PEXOCRenderSingle or PEXOCStoreSingle).
op The model clipping volume operator (PEXModelClipReplace or PEX-
ModelClipIntersection).
count The number of halfspaces.
half_spaces An array of points and normal vectors defining the model clipping
volume.

RETURNS None

DESCRIPTION This function creates an output primitive attribute which sets the model clip volume attri-
but. The operator will be used to combine the specified list of half-spaces with the
current model clipping volume to form a new model clipping volume. Values for the
operator are PEXModelClipReplace (replace current volume) and PEXModelClipInter-
section (intersect specified volume with current volume). Each half-space is defined by a
point and a normal in model coordinates. The normal points in the direction of the half-
space, and the point is considered to be on the plane. See PEXGetImpDepConstants(3)
for the maximum allowable number of model clip planes.
Each half-space is transformed by the current composite modeling transformation and
combined with the current model clipping volume. The resulting model clipping volume
is not affected by subsequent changes to the composite modeling transformation.

DATA STRUCTURES
typedef struct {
    PEXCoord point;
    PEXVector vector;
} PEXHalfSpace;
See also PEXlib.h.

ERRORS
BadPEXRenderer
The specified renderer resource identifier is invalid.
BadPEXStructure
The specified structure resource identifier is invalid.
| SEE ALSO | PEXSetModelClipFlag(3)  
|          | PEXSetModelClipVolume2D(3)  
|          | PEXRestoreModelClipVolume(3)  
|          | PEXGetImpDepConstants(3)  |
NAME | PEXSetModelClipVolume2D - Set Model Clip Volume 2D  
SYNTAX | void PEXSetModelClipVolume2D(Display *display, XID resource_id, PEXOCRequestType req_type, int op, unsigned int count, PEXHalfSpace2D *half_spaces)  
PARAMETERS | display | A pointer to a display structure returned by a successful XOpenDisplay call.  
resource_id | The resource identifier of the renderer or structure.  
req_type | The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).  
op | The model clipping volume operator (PEXModelClipReplace or PEXModelClipIntersection).  
count | The number of halfspaces.  
half_spaces | An array of points and normal vectors defining the model clipping volume.  
RETURNS | None  
DESCRIPTION | This function creates an output primitive attribute which sets the model clip volume attribute. This output command is similar to PEXSetModelClipVolume(3) except that the half-spaces are specified in 2D coordinates where the z-component of each point and normal vector assumed to be zero.  
DATA STRUCTURES | typedef struct {  
PEXCoord2D point;  
PEXVector2D vector;  
} PEXHalfSpace2D;  
See also PEXlib.h.  
ERRORS | BadPEXRenderer  
The specified renderer resource identifier is invalid.  
BadPEXStructure  
The specified structure resource identifier is invalid.  
SEE ALSO | PEXSetModelClipFlag(3)  
PEXSetModelClipVolume(3)  
PEXRestoreModelClipVolume(3)  
PEXGetImpDepConstants(3)  
modified November 1995
**NAME**
PEXSetOfFillAreaSets - 3D Set of Fill Area Sets Primitive

**SYNTAX**
void PEXSetOfFillAreaSets(Display *display, XID resource_id, PEXOCRequestType req_type, int shape_hint, unsigned int facet_attributes, unsigned int vertex_attributes, unsigned int edge_attributes, int contour_hint, int contours_all_one, int color_type, unsigned int set_count, PEXArrayOfFacetData facet_data, unsigned int vertex_count, PEXArrayOfVertex vertices, unsigned int index_count, PEXSwitch *edge_flags, PEXConnectivityData *connectivity)

**PARAMETERS**
display  A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id  The resource identifier of the renderer or structure.
req_type  The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
shape_hint  The shape which describes all of the contours (PEXShapeComplex, PEXShapeNonConvex, PEXShapeConvex, PEXShapeUnknown).
facet_attributes  A mask indicating the facet attributes provided (PEXGANone, PEXGAColor, PEXGANormal).
vertex_attributes  A mask indicating the vertex attributes provided (PEXGANone, PEXGAColor, PEXGANormal).
edge_attributes  A mask indicating the edge attributes provided (PEXGANone or PEXGAEdges).
contour_hint  A flag indicating whether contours are disjoint or overlapping (PEXContourDisjoint, PEXContourNested, PEXContourIntersecting, PEXContourUnknown).
contours_all_one  True if each fill area set contains only one contour; False otherwise.
color_type  The type of color data provided (PEXColorTypeIndexed, PEXColorTypeRGB, PEXColorTypeCIE, PEXColorTypeHSV, PEXColorTypeHLS, PEXColorTypeRGB8, PEXColorTypeRGB16).
set_count  The number of fill area sets.
facet_data  An array of facet data.
vertex_count  The number of vertices.
vertices  An array of vertices.
index_count  The number of vertex connectivity indices (also number of edge flags, if edges are specified).
edge_flags  An array of edge flags.
connectivity  A pointer to the list of contour connectivity data.

342 modified November 1995
RETURNS
None

DESCRIPTION
This function creates a set of fill area sets output primitive.

A fill area may cross over itself to create a complex shape. The odd-even rule is used for determining the area that lies in the interior of the fill area. The shape hint is provided to enable performance improvements for certain shapes. Fill areas that are of higher complexity than indicated by the shape hint are rendered in an implementation-dependent manner. Consequently, applications should pass PEXShapeUnknown as the shape unless they are certain the fill area’s shape is one of the other three. Note that a fill area set with more than one contour is always allowed to have contours that intersect. If is quite possible that the only times rendering optimization can occur are when the number of contours in a fill area set is equal to one or if the contours all one flag is True, and the shape hint is PEXShapeConvex.

The contour hint provides further information about the relationships between contours in the fill area set. If the contour hint is PEXContourDisjoint all contours will be spatially disjoint. No overlapping or intersection occurs between any contours in the fill area set. If the contour hint is PEXContourNested contours will either be disjoint or wholly contained within another contour. No contour will have edges that intersect or are coincident with edges of any other contour. If the contour hint is PEXContourIntersecting separated contours may have edges that are coincident or overlap. If the contour hint is PEXContourUnknown nothing is known about the interrelationships between contours. Fill area sets with contours that have higher complexity interrelationships than that indicated by the contour hint are rendered in an implementation-dependent manner.

The facet attributes indicate the content of the facet data. This data may be a color, a normal, or a color followed by a normal. Use the constants PEXGANone, PEXGAColor and PEXGANormal to construct a mask indicating the data provided. If specified, the facet color takes precedence over the surface color. If specified, the facet normal is used to determine whether the fill area is back-facing.

The vertex attributes indicate the content of each fill area vertex. In addition to the coordinate (x,y,z), applications may specify a color, a normal, or a color followed by a normal for each vertex. Use the constants PEXGANone, PEXGAColor and PEXGANormal to construct a mask indicating the data provided. If specified, vertex colors will override facet color or the current surface color. If specified, vertex normals are taken to be normals at the vertices of the fill area.

The reflection model and the surface interpolation will affect how the additional data is used in rendering the surface.

Color values passed must be of the specified color type. Normals are assumed to be unit length vectors. The effect if the normal is not unit length is implementation-dependent.

The edge attributes indicate the content of the edge flags. Use PEXGAEdges to indicate edge flags are provided or PEXGANone if no edge flags are provided. The edge flags, if present, are set to PEXOn or PEXOff and are used to indicate which edges should be rendered. The edge control for vertex i indicates whether or not to render the edge between vertex i and vertex i+1. Surface edges are always rendered with the surface edge color.
and are not affected by the facet or vertex colors.

The connectivity of the primitive is defined by the connectivity list. The connectivity list is a pointer to an array of PEXConnectivityData structures. Each entry in the array gives the contours for one fill area set in the set of fill area sets, and, in turn, contains a pointer to an array of PEXListOfUShort structures. Each of these latter structures gives the index of the vertices of one contour in that fill area set. The indices select a vertex in the array of vertices. Vertices are numbered with indices starting from zero (i.e. the first vertex is referenced as vertex 0). As a special case, if the contours all one flag is True then the contour count field in each fill area set is guaranteed to be one.

All attributes affecting the representation of fill area sets also affect the representation of this primitive.

**DATA STRUCTURES**

```c
typedef struct {
    unsigned short count; /* number of lists */
    PEXListOfUShort *lists;
} PEXConnectivityData;

typedef struct {
    unsigned short count; /* number of shorts */
    unsigned short *shorts;
} PEXListOfUShort;

typedef unsigned char PEXSwitch;
```

See also PEXlib.h.

**ERRORS**

- **BadPEXOutputCommand**
  The output command contains an invalid value.
- **BadPEXRenderer**
  The specified renderer resource identifier is invalid.
- **BadPEXStructure**
  The specified structure resource identifier is invalid.

**SEE ALSO**

- PEXSetInteriorStyle(3)
- PEXSetInteriorStyleIndex(3)
- PEXSetSurfaceColorIndex(3)
- PEXSetSurfaceColor(3)
- PEXSetReflectionAttributes(3)
- PEXSetReflectionModel(3)
- PEXSetSurfaceInterpMethod(3)
- PEXSetBFImpInteriorStyle(3)
- PEXSetBFImpInteriorStyleIndex(3)
- PEXSetBFSurfaceColorIndex(3)
- PEXSetBFSurfaceColor(3)
- PEXSetBFReflectionAttributes(3)
PEXSetBFReflectionModel(3)
PEXSetBFSurfaceInterpMethod(3)
PEXSetFacetCullingMode(3)
PEXSetFacetDistinguishFlag(3)
PEXSetPatternSize(3)
PEXSetPatternAttributes(3)
PEXSetPatternAttributes2D(3)
PEXSetInteriorBundleIndex(3)
PEXSetSurfaceEdgeFlag(3)
PEXSetSurfaceEdgeType(3)
PEXSetSurfaceEdgeWidth(3)
PEXSetSurfaceEdgeColor(3)
PEXSetSurfaceEdgeColorIndex(3)
PEXSetEdgeBundleIndex(3)
PEXSetPCAttributeMask - Macro to Setup Pipeline Context Attributes Value Mask

**SYNTAX**

```
PEXSetPCAttributeMask(mask, attr)
```

**PARAMETERS**

- **mask**
  The address of the value mask - an array of three unsigned long.

- **attr**
  A single pipeline context attribute bitmask constant.

**DESCRIPTION**

This is a utility macro to aid in setting up the bitmask for the pipeline context attributes. The following attribute bitmask constants must be used:

```
PEXPCASFValues PEXPCLocalTransform
PEXPCATextAlignment PEXPCEmarkerBundleIndex
PEXPCATextHeight PEXPCEmarkerColor
PEXPCATextPath PEXPCEmarkerScale
PEXPCATextStyle PEXPCEmarkerType
PEXPCATextUpVector PEXPCEmarkerClip
PEXPCBFInteriorStyle PEXPCEmarkerClipVolume
PEXPCBFInteriorStyleIndex PEXPCEmarkerClipVolume
PEXPCBFReflectionAttr PEXPCEmarkerClipVolume
PEXPCBFReflectionModel PEXPCEmarkerClipVolume
PEXPCBFSurfaceColor PEXPCEmarkerClipVolume
PEXPCBSurfaceInterp PEXPCEmarkerClipVolume
PEXPCBSurfaceVolume PEXPCEmarkerClipVolume
PEXPCBCharExpansion PEXPCEmarkerClipVolume
PEXPCBCharHeight PEXPCEmarkerClipVolume
PEXPCBCharSpacing PEXPCEmarkerClipVolume
PEXPCBCharUpVector PEXPCEmarkerClipVolume
PEXPCColorApproxIndex PEXPCEmarkerClipVolume
PEXPCCullingMode PEXPCEmarkerClipVolume
PEXPCCurvApprox PEXPCEmarkerClipVolume
PEXPCCDepthCueIndex PEXPCEmarkerClipVolume
PEXPCDistinguishFlag PEXPCEmarkerClipVolume
PEXPCEdgeBundleIndex PEXPCEmarkerClipVolume
PEXPCEGlobalTransform PEXPCEmarkerClipVolume
PEXPCHELSRIentifier PEXPCEmarkerClipVolume
PEXPCEInteriorBundleIndex PEXPCEmarkerClipVolume
PEXPCEInteriorStyle PEXPCEmarkerClipVolume
PEXPCEInteriorStyleIndex PEXPCEmarkerClipVolume
PEXPCLightState PEXPCEmarkerClipVolume
PEXPCLineBundleIndex PEXPCEmarkerClipVolume
PEXPCLLineColor PEXPCEmarkerClipVolume
PEXPCLLineType PEXPCEmarkerClipVolume
PEXPCLineWidth PEXPCEmarkerClipVolume
```

Note that this macro does multiple evaluations of the value for attr.
<table>
<thead>
<tr>
<th>ERRORS</th>
<th>None</th>
</tr>
</thead>
</table>
| SEE ALSO| PEXCreatePipelineContext(3)  
          | PEXChangePipelineContext(3)  
          | PEXCopyPipelineContext(3)    
<pre><code>      | PEXGetPipelineContext(3)     |
</code></pre>
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXSetPCAttributeMaskAll - Macro to Set All Pipeline Context Attributes in Value Mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td>PEXSetPCAttributeMaskAll(mask)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>mask The address of the value mask - an array of three unsigned long.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>This is a utility macro to aid in setting up the bitmask for the pipeline context attributes. This macro will set all valid bits in the mask.</td>
</tr>
<tr>
<td>ERRORS</td>
<td>None</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>PEXCreatePipelineContext(3) PEXChangePipelineContext(3) PEXCopyPipelineContext(3) PEXGetPipelineContext(3)</td>
</tr>
</tbody>
</table>
NAME
PEXSetPWAttributeMask - Macro to Setup Workstation Attributes Value Mask

SYNTAX
PEXSetPWAttributeMask(mask, attr)

PARAMETERS
mask The address of the value mask - an array of two unsigned long.
attr A single workstation attribute bitmask constant.

DESCRIPTION
This is a utility macro to aid in setting up the bitmask for the workstation attributes. The following constants must be used:

PEXPWBufferUpdate PEXPWInvisibilityExcl
PEXPWColorApproxTable PEXPWInvisibilityIncl
PEXPWColorTable PEXPWLightTable
PEXPWCurBufferMode PEXPWLineBundle
PEXPWCurHLHSRMode PEXPWMarkerBundle
PEXPWCurNPCSubVolume PEXPWNumPriorities
PEXPWCurViewport PEXPWPToolBarTable
PEXPWDefinedViews PEXPWPostedStructures
PEXPWDepthCueTable PEXPWReqBufferMode
PEXPWDisplaySurface PEXPWReqHLHSRMode
PEXPWDisplayUpdate PEXPWReqNPCSubVolume
PEXPWDrawable PEXPWReqViewport
PEXPWEdgeBundle PEXPWTextBundle
PEXPWHLHSRUpdate PEXPWTextFontTable
PEXPWHighlightExcl PEXPWViewUpdate
PEXPWHighlightIncl PEXPWVisualState
PEXPWInteriorBundle PEXPWWorkstationUpdate

Note that this macro does multiple evaluations of the value for attr.

ERRORS
None

SEE ALSO PEXGetWorkstationAttributes(3)

modified November 1995
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXSetPWAtributeMaskAll - Macro to Set All Workstation Attributes in Value Mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td>PEXSetPWAtributeMaskAll(mask)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>mask The address of the value mask - an array of two unsigned long.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>This is a utility macro to aid in setting up the bitmask for the workstation attributes. This macro will set all valid bits in the mask.</td>
</tr>
<tr>
<td>ERRORS</td>
<td>None</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>PEXGetWorkstationAttributes(3)</td>
</tr>
</tbody>
</table>
NAME
PEXSetParaSurfCharacteristics - Set Parametric Surface Characteristics

SYNTAX
void PEXSetParaSurfCharacteristics(Display *display, XID resource_id, PEXOCRequestType req_type, int psc_type, PEXPSCData *characteristics)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
psc_type The parametric surface characteristic type (PEXPSCNone, PEXPSCImpDep, PEXPSCIsoCurves, PEXPSCMCLevelCurves, PEXPSCWCLevelCurves).
characteristics A pointer to data defining the parametric surface characteristics.

RETURNS
None

DESCRIPTION
This function creates an output primitive attribute which sets the parametric surface characteristics. If the specified parametric surface characteristics type is not supported, PEXPSCNone is used. Supported values for parametric surface characteristic types are inquirable via PEXGetEnumTypeInfo(3).

DATA STRUCTURES
typedef union {
    PEXPSCIsoparametricCurves iso_curves;
    PEXPSCLevelCurves level_curves;
    PEXPSCImpDepData imp_dep;
} PEXPSCData;

typedef struct {
    unsigned short placement_type;
    unsigned short reserved;
    unsigned short u_count;
    unsigned short v_count;
} PEXPSCIsoparametricCurves;

typedef struct {
    PEXCoord origin;
    PEXVector direction;
    unsigned short count; /* number of parameters */
    unsigned short reserved;
    float *parameters;
} PEXPSCLevelCurves;

typedef struct {

modified November 1995 351
unsigned short length;
char *data;
}

} PEXPSCImpDepData;

See also PEXlib.h.

ERRORS

BadPEXOutputCommand
The output command contains an invalid value.

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.
### NAME
PEXSetPatternAttributes - Set Pattern Interior Style Attributes

### SYNTAX
```c
void PEXSetPatternAttributes(Display *display, XID resource_id, PEXOCRequestType req_type, PEXCoord *ref_point, PEXVector *vector1, PEXVector *vector2)
```

### PARAMETERS
- **display**: A pointer to a display structure returned by a successful `XOpenDisplay` call.
- **resource_id**: The resource identifier of the renderer or structure.
- **req_type**: The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- **ref_point**: The pattern reference point.
- **vector1**: The first pattern reference vector.
- **vector2**: The second pattern reference vector.

### RETURNS
None

### DESCRIPTION
This function creates an output primitive attribute which sets the pattern reference point and both pattern reference vectors attributes, in modeling coordinates. If the surface interior style is PEXInteriorStylePattern, the pattern reference point and the pattern reference vectors are used to position and scale the pattern on the surface. If either of the pattern reference vectors are zero length or if the vectors are parallel, the output command is ignored.

### DATA STRUCTURES
See PEXlib.h.

### ERRORS
- **BadPEXRenderer**: The specified renderer resource identifier is invalid.
- **BadPEXStructure**: The specified structure resource identifier is invalid.
### NAME
PEXSetPatternAttributes2D - Set Pattern Interior Style Reference Point

### SYNTAX
```c
void PEXSetPatternAttributes2D(Display *display, XID resource_id, PEXOCRequestType req_type, PEXCoord2D *ref_point)
```

### PARAMETERS
- **display**: A pointer to a display structure returned by a successful `XOpenDisplay` call.
- **resource_id**: The resource identifier of the renderer or structure.
- **req_type**: The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- **ref_point**: The pattern reference point.

### RETURNS
None

### DESCRIPTION
This function creates an output primitive attribute which sets the pattern reference point and both pattern reference vector attributes, in 2D modeling coordinates. The pattern reference point is set to \( <x, y, 0> \) and the two pattern reference vectors are set to \( <1, 0, 0> \) and \( <0, 1, 0> \). If the surface interior style is `PEXInteriorStylePattern`, the pattern reference point and the reference vectors are used to position and scale the pattern on the surface.

### DATA STRUCTURES
See PEXlib.h.

### ERRORS
- **BadPEXRenderer**: The specified renderer resource identifier is invalid.
- **BadPEXStructure**: The specified structure resource identifier is invalid.
**NAME**  
PEXSetPatternSize - Set Pattern Interior Style Size

**SYNTAX**  
void PEXSetPatternSize(Display *display, XID resource_id, PEXOCRequestType req_type, double width, double height)

**PARAMETERS**  
- **display**  
  A pointer to a display structure returned by a successful XOpenDisplay call.
- **resource_id**  
  The resource identifier of the renderer or structure.
- **req_type**  
  The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- **width**  
  The pattern width.
- **height**  
  The pattern height.

**RETURNS**  
None

**DESCRIPTION**  
This function creates an output primitive attribute which sets the pattern size attribute. Only the magnitude of the specified width and height is used. The value (width,0) is used as the pattern width vector, and the value (0,height) specifies the pattern height vector. If the interior style is PEXInteriorStylePattern, these values plus the pattern reference point and the pattern reference vector are used to position, scale, and rotate the pattern on the surface.

**ERRORS**  
- **BadPEXRenderer**  
  The specified renderer resource identifier is invalid.
- **BadPEXStructure**  
  The specified structure resource identifier is invalid.
NAME PEXSetPickID - Set Pick Identifier

SYNTAX void PEXSetPickID(Display *display, XID resource_id, PEXOCRequestType req_type,
                           unsigned long pick_id)

PARAMETERS

  display A pointer to a display structure returned by a successful XOpenDisplay call.
  resource_id The resource identifier of the renderer or structure.
  req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRendersonSingle or PEXOCStoreSingle).
  pick_id The pick identifier.

RETURNS None

DESCRIPTION This function creates an output primitive attribute which sets the current pick identifier attribute.

ERRORS

BadPEXRenderer The specified renderer resource identifier is invalid.

BadPEXStructure The specified structure resource identifier is invalid.
NAME PEXSetPolylineInterpMethod - Set Polyline Interpolation Method

SYNTAX void PEXSetPolylineInterpMethod(Display *display, XID resource_id, PEXOCRequestType req_type, int method)

PARAMETERS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>display</td>
<td>A pointer to a display structure returned by a successful XOpenDisplay call.</td>
</tr>
<tr>
<td>resource_id</td>
<td>The resource identifier of the renderer or structure.</td>
</tr>
<tr>
<td>req_type</td>
<td>The request type for the output command ( PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle ).</td>
</tr>
<tr>
<td>method</td>
<td>The polyline interpolation method ( PEXPolylineInterpNone or PEXPolylineInterpColor ).</td>
</tr>
</tbody>
</table>

RETURNS None

DESCRIPTION This function creates an output primitive attribute which sets the polyline interpolation method attribute. If the specified interpolation method is not supported, PEXPolylineInterpNone is used. Supported values for polyline interpolation method are inquirable via PEXGetEnumTypeInfo(3).

ERRORS BadPEXRenderer

The specified renderer resource identifier is invalid.

BadPEXStructure

The specified structure resource identifier is invalid.

SEE ALSO PEXGetEnumTypeInfo(3)
PEXSetIndividualASF(3)
NAME | PEXSetReflectionAttributes - Set Surface Reflection Attributes  
SYNTAX | void PEXSetReflectionAttributes(Display *display, XID resource_id, PEXOCRequestType req_type, PEXReflectionAttributes *attributes)  
PARAMETERS |  
| display | A pointer to a display structure returned by a successful XOpenDisplay call.  
| resource_id | The resource identifier of the renderer or structure.  
| req_type | The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).  
| attributes | The surface reflection attributes.  
RETURNS | None  
DESCRIPTION | This function creates an output primitive attribute which sets the surface reflection attributes. Surface reflection attributes consist of the ambient, diffuse and specular coefficients, the specular concentration and color, and the transmission coefficient.  
DATA STRUCTURES |  
| typedef struct {  
| float | ambient;  
| float | diffuse;  
| float | specular;  
| float | specular_conc;  
| float | transmission;  
| PEXColorSpecifier | specular_color;  
| } PEXReflectionAttributes;  
See also PEXlib.h.  
ERRORS |  
| BadPEXRenderer | The specified renderer resource identifier is invalid.  
| BadPEXStructure | The specified structure resource identifier is invalid.  
SEE ALSO | PEXSetIndividualASF(3)  
PEXGetImpDepConstants(3)  
358 modified November 1995
| NAME | PEXSetReflectionModel - Set Surface Reflection Model |
| SYNTAX | void PEXSetReflectionModel( Display *display, XID resource_id, PEXOCRequestType req_type, int model) |
| PARAMETERS | display | A pointer to a display structure returned by a successful XOpenDisplay call. |
| | resource_id | The resource identifier of the renderer or structure. |
| | req_type | The request type for the output command (PEXOCRender, PEXOCRenderSingle, PEXOCStore, or PEXOCStoreSingle). |
| | model | The surface reflection model (PEXReflectionNone, PEXReflectionAmbient, PEXReflectionDiffuse, or PEXReflectionSpecular). |
| RETURNS | None |
| DESCRIPTION | This function creates an output primitive attribute which sets the surface reflection model attribute. If the specified reflection model is not supported, PEXReflectionNone is used. Supported values for surface reflection model are inquirable via PEXGetEnumTypeInfo(3). |
| ERRORS | BadPEXRenderer | The specified renderer resource identifier is invalid. |
| | BadPEXStructure | The specified structure resource identifier is invalid. |
| SEE ALSO | PEXGetEnumTypeInfo(3) |
| | PEXSetIndividualASF(3) |
NAME  PEXSetRenderingColorModel - Set Rendering Color Model

SYNTAX  void PEXSetRenderingColorModel(Display *display, XID resource_id, PEXOCRequestType req_type, int model)

PARAMETERS  display  A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id  The resource identifier of the renderer or structure.
req_type  The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
model  The surface reflection model (PEXRenderingColorModelImpDep, PEXRenderingColorModelRGB, PEXRenderingColorModelCIE, PEXRenderingColorModelHSV, PEXRenderingColorModelHLS).

RETURNS  None

DESCRIPTION  This function creates an output primitive attribute which sets the rendering color model attribute. If the specified color model is not supported, an implementation-dependent model (model 0) is used. Supported values for rendering color model are inquirable via PEXGetEnumTypeInfo.

ERRORS  BadPEXRenderer
The specified renderer resource identifier is invalid.
BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO  PEXGetEnumTypeInfo
NAME
PEXSetSurfaceApprox - Set Surface Approximation Method

SYNTAX
void PEXSetSurfaceApprox(Display *display, XID resource_id, PEXOCRequestType
req_type, int method, double utolerance, double vtolerance)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay
call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOC-
Store, PEXOCRenderSingle or PEXOCStoreSingle).
method The surface approximation method (PEXApproxImpDep, PEXApprox-
ConstantBetweenKnots, PEXApproxWCChordalSize, PEXApproxNPCChordalSize,
PEXSurfaceApproxWCPlanarDev, PEXSurfaceApproxNPCPlanarDev, PEXSur-
faceApproxDCPlanarDev, PEXApproxWCRelative, PEXApproxNPCRela-
tive, PEXApproxDCRelative).

utolerance The surface approximation tolerance in the u direction.
vtolerance The surface approximation tolerance in the v direction.

RETURNS None

DESCRIPTION
This function creates an output primitive attribute which sets the surface approximation
method attribute. If the specified method is not supported, PEXApproxImpDep is used.
Supported values for surface approximation are inquirable via PEXGetEnum-
TypeInfo(3). The u and v tolerance values are provided to indicate the desired accuracy
of the approximation, and are used in different ways for the different methods.

ERRORS
BadPEXRenderer
The specified renderer resource identifier is invalid.
BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO
PEXSetIndividualASF(3)
PEXGetEnumTypeInfo(3)

modified November 1995
NAME
PEXSetSurfaceColor - Set Surface Color

SYNTAX
void PEXSetSurfaceColor(Display *display, XID resource_id, PEXOCRequestType
req_type, int color_type, PEXColor *color)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay
      call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOC-
      Store, PEXOCRenderSingle or PEXOCStoreSingle).
color_type The type of color (PEXColorTypeIndexed, PEXColorTypeRGB, PEX-
      ColorTypeCIE, PEXColorTypeHSV, PEXColorTypeHLS, PEXColor-
      TypeRGB8, PEXColorTypeRGB16).
color A pointer to the surface color.

RETURNS
None

DESCRIPTION
This function creates an output primitive attribute which sets the surface color attribute.
The attribute is set to either an indexed color or a direct color value, depending on the
color type. If the color type is PEXColorTypeIndexed and the specified color index is not
defined, color index one is used. If color index one is not defined, the result is
implementation-dependent.

DATA STRUCTURES
See PEXlib.h.

ERRORS
BadPEXOutputCommand
The output command contains an invalid value, e.g. the color type is PEXColor-
TypeIndexed, and the color index value exceeds 65534.

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO
PEXSetIndividualASF(3)
PEXCreateLookupTable(3)

362 modified November 1995
NAME
PEXSetSurfaceColorIndex - Set Surface Color Index

SYNTAX
void PEXSetSurfaceColorIndex(Display *display, XID resource_id, PEXOCRequestType req_type, unsigned int index)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
index The color table index surfaces.

RETURNS None

DESCRIPTION
This function creates an output primitive attribute which sets the surface color attribute to an indexed color with the value indicated by index. If the specified color index is not defined, color index one is used. If color index one is not defined, the result is implementation-dependent.

ERRORS
BadPEXOutputCommand
The color index value exceeds 65534.

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO
PEXSetIndividualASF(3)
PEXCreateLookupTable(3)
NAME
PEXSetSurfaceEdgeColor - Set Surface Edge Color

SYNTAX
void PEXSetSurfaceEdgeColor(Display *display, XID resource_id, PEXOCRequestType req_type, int color_type, PEXColor *color)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
color_type The type of color (PEXColorTypeIndexed, PEXColorTypeRGB, PEXColorTypeCIE, PEXColorTypeHSV, PEXColorTypeHLS, PEXColorTypeRGB8, PEXColorTypeRGB16).
color A pointer to the surface edge color.

RETURNS
None

DESCRIPTION
This function creates an output primitive attribute which sets the surface edge color attribute. The attribute is set to either an indexed color or a direct color value, depending on the color type. If the color type is PEXColorTypeIndexed and the specified color index is not defined, color index one is used. If color index one is not defined, the result is implementation-dependent.

DATA STRUCTURES
See PEXlib.h.

ERRORS
BadPEXOutputCommand
The output command contains an invalid value, e.g. the color type is PEXColorTypeIndexed, and the color index value exceeds 65534.

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO
PEXSetIndividualASF(3)
PEXCreateLookupTable(3)

modified November 1995
NAME
PEXSetSurfaceEdgeColorIndex - Set Surface Edge Color Index

SYNTAX
void PEXSetSurfaceEdgeColorIndex(Display *display, XID resource_id, PEXOCRequestType req_type, unsigned int index)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
index The color table index for surface edges.

RETURNS
None

DESCRIPTION
This function creates an output primitive attribute which sets the surface edge color attribute to an indexed color with the value indicated by index. If the specified color index is not defined, color index one is used. If color index one is not defined, the result is implementation-dependent.

ERRORS
BadPEXOutputCommand
The color index value exceeds 65534.
BadPEXRenderer
The specified renderer resource identifier is invalid.
BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO
PEXSetIndividualASF(3)
PEXCreateLookupTable(3)

modified November 1995
**NAME**
PEXSetSurfaceEdgeFlag - Set Surface Edge Flag

**SYNTAX**
```c
void PEXSetSurfaceEdgeFlag(Display *display, XID resource_id, PEXOCRequestType req_type, int flag)
```

**PARAMETERS**
- **display**
  A pointer to a display structure returned by a successful `XOpenDisplay` call.
- **resource_id**
  The resource identifier of the renderer or structure.
- **req_type**
  The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- **flag**
  A flag indicating whether surface edge drawing is enabled or disabled (PEXOn or PEXOff).

**RETURNS**
None

**DESCRIPTION**
This function creates an output primitive attribute which sets the surface edge flag attribute. Values for the surface edge flag are PEXOn (enable surface edges) and PEXOff (disable surface edges).

**ERRORS**
- **BadPEXOutputCommand**
  The output command contains an invalid value.
- **BadPEXRenderer**
  The specified renderer resource identifier is invalid.
- **BadPEXStructure**
  The specified structure resource identifier is invalid.

**SEE ALSO**
PEXSetIndividualASF(3)
<table>
<thead>
<tr>
<th><strong>NAME</strong></th>
<th>PEXSetSurfaceEdgeType - Set Surface Edge Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SYNTAX</strong></td>
<td>void PEXSetSurfaceEdgeType(Display *display, XID resource_id, PEXOCRequestType req_type, int edge_type)</td>
</tr>
<tr>
<td><strong>PARAMETERS</strong></td>
<td></td>
</tr>
<tr>
<td>display</td>
<td>A pointer to a display structure returned by a successful XOpenDisplay call.</td>
</tr>
<tr>
<td>resource_id</td>
<td>The resource identifier of the renderer or structure.</td>
</tr>
<tr>
<td>req_type</td>
<td>The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).</td>
</tr>
<tr>
<td>edge_type</td>
<td>The surface edge type (PEXSurfaceEdgeSolid, PEXSurfaceEdgeDotted, PEXSurfaceEdgeDashed, PEXSurfaceEdgeDashDot).</td>
</tr>
<tr>
<td><strong>RETURNS</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>DESCRIPTION</strong></td>
<td>This function creates an output primitive attribute which sets the surface edge type attribute. If the specified edge type is not supported, PEXSurfaceEdgeSolid is used. Supported values for surface edge type are inquirable via PEXGetEnumTypeInfo(3).</td>
</tr>
<tr>
<td><strong>ERRORS</strong></td>
<td></td>
</tr>
<tr>
<td>BadPEXRenderer</td>
<td>The specified renderer resource identifier is invalid.</td>
</tr>
<tr>
<td>BadPEXStructure</td>
<td>The specified structure resource identifier is invalid.</td>
</tr>
<tr>
<td><strong>SEE ALSO</strong></td>
<td>PEXGetEnumTypeInfo(3) PEXSetIndividualASF(3)</td>
</tr>
</tbody>
</table>

modified November 1995
NAME
PEXSetSurfaceEdgeWidth - Set Surface Edge Width

SYNTAX
void PEXSetSurfaceEdgeWidth(Display *display, XID resource_id, PEXOCRequestType req_type, double width)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
width The surface edge width scale factor.

RETURNS None

DESCRIPTION
This function creates an output primitive attribute which sets the surface edge width attribute. The specified width is used as a scale factor. This scale factor is used to increase or decrease the width (in pixels) from the nominal edge width for the display device. The result is mapped to the nearest supported line width.

ERRORS
BadPEXRenderer
The specified renderer resource identifier is invalid.
BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO
PEXGetImpDepConstants(3)
PEXSetIndividualASF(3)
NAME
PEXSetSurfaceInterpMethod - Set Surface Interpolation Method

SYNTAX
void PEXSetSurfaceInterpMethod(Display *display, XID resource_id, PEXOCRequestType req_type, int method)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
method The surface interpolation method (PEXSurfaceInterpNone, PEXSurfaceInterpColor, PEXSurfaceInterpDotProduct, PEXSurfaceInterpNormal).

RETURNS
None

DESCRIPTION
This function creates an output primitive attribute which sets the surface interpolation method attribute. If the specified interpolation method is not supported, PEXSurfaceInterpNone is used. Supported values for surface interpolation are inquirable via PEXGetEnumTypeInfo(3).

ERRORS
BadPEXRenderer
The specified renderer resource identifier is invalid.
BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO
PEXGetEnumTypeInfo(3)
PEXSetIndividualASF(3)

modified November 1995
PEXSetTableEntries - Set Lookup Table Entries

void PEXSetTableEntries(Display *display, PEXLookupTable table, unsigned int start, unsigned int count, int table_type, PEXPointer entries)

PARAMETERS

display A pointer to a display structure returned by a successful XOpenDisplay call.
table The resource identifier of the lookup table.
start The index of the first table entry to be set.
count The number of table entries to be set.
type The type of lookup table entries to be set (see the Description).
entries An array of table entries.

RETURNS None

DESCRIPTION

This function sets lookup table entries in the specified lookup table, starting at the specified entry index. The entries must point to an array of structures having one of the following types:

PEXTextFontEntry if type is PEXLUTTextFont
PEXViewEntry if type is PEXLUTView
PEXColorApproxEntry if type is PEXLUTColorApprox
PEXLineBundleEntry if type is PEXLUTLineBundle
PEXMarkerBundleEntry if type is PEXLUTMarkerBundle
PEXTextBundleEntry if type is PEXLUTTextBundle
PEXInteriorBundleEntry if type is PEXLUTInteriorBundle
PEXEdgeBundleEntry if type is PEXLUTEdgeBundle
PEXLightEntry if type is PEXLUTLight
PEXDepthCueEntry if type is PEXLUTDepthCue
PEXColorEntry if type is PEXLUTColor
PEXPatterEntry if type is PEXLUTPattern

DATA STRUCTURES

typedef XID PEXLookupTable;

#if NeedFunctionPrototypes
typedef void *PEXPointer;
#else
typedef char *PEXPointer;
#endif

ERRORS

BadAlloc
The server failed to allocate the resource.

BadPEXColorType
The specified color type is invalid or unsupported.
BadPEXLookupTable
The specified lookup table resource identifier is invalid, or the table type is unsupported.

BadValue
The sum of start plus count is too large, a table entry field contains an invalid value, or index 0 is invalid for the specified table type.

SEE ALSO
PEXCreateLookupTable(3)
PEXGetTableInfo(3)
PEXGetPredefinedEntries(3)
PEXGetDefinedIndices(3)
PEXGetTableEntry(3)
PEXGetTableEntries(3)
NAME

PEXSetTextAlignment - Set Text Alignment

SYNTAX

void PEXSetTextAlignment(Display *display, XID resource_id, PEXOCRequestType
req_type, int halignment, int valignment)

PARAMETERS

display A pointer to a display structure returned by a successful XOpenDisplay
 call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOC-
Store, PEXOCRenderSingle or PEXOCStoreSingle).
halignment The horizontal text alignment (PEXHAlignNormal, PEXHAlignLeft,
PEXHAlignCenter, PEXHAlignRight).
valignment The vertical text alignment (PEXVAlignNormal, PEXVAlignTop,
PEXVAlignCap, PEXVAlignHalf, PEXVAlignBase, PEXVAlignBottom).

RETURNS

None

DESCRIPTION

This function creates an output primitive attribute which sets the horizontal and vertical
text alignment attributes.

ERRORS

BadPEXOutputCommand
The output command contains an invalid value.
BadPEXRenderer
The specified renderer resource identifier is invalid.
BadPEXStructure
The specified structure resource identifier is invalid.
NAME
PEXSetTextBundleIndex - Set Text Bundle Index

SYNTAX
void PEXSetTextBundleIndex(Display *display, XID resource_id, PEXOCRequestType req_type, unsigned int index)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
index The text bundle table index.

RETURNS None

DESCRIPTION
This function creates an output primitive attribute which sets the text bundle index attribute. If an undefined index is specified, the default bundle index one is used. If index one is undefined, the defaults are used: text font index is one; text precision is PEXStringPrecision; character expansion is 1.0; character spacing is 0.0 and text color is indexed color one.

A text bundle table index of 0 will produce a BadPEXOutputCommand error.

ERRORS
BadPEXOutputCommand
The output command contains an invalid value.

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO
PEXSetIndividualASF(3)
PEXCreateLookupTable(3)

modified November 1995
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXSetTextColor - Set Text Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td>void PEXSetTextColor(Dispaly ∗display, XID resource_id, PEXOCRequestType req_type, int color_type, PEXColor ∗color)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>display</td>
</tr>
<tr>
<td></td>
<td>resource_id</td>
</tr>
<tr>
<td></td>
<td>req_type</td>
</tr>
<tr>
<td></td>
<td>color_type</td>
</tr>
<tr>
<td></td>
<td>color</td>
</tr>
<tr>
<td>RETURNS</td>
<td>None</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>This function creates an output primitive attribute which sets the text color attribute. The attribute is set to either an indexed color or a direct color value, depending on the color type. If the color type is PEXColorTypeIndexed and the specified color index is not defined, color index one is used. If color index one is not defined, the result is implementation-dependent.</td>
</tr>
<tr>
<td>DATA STRUCTURES</td>
<td>See PEXlib.h.</td>
</tr>
<tr>
<td>ERRORS</td>
<td>BadPEXOutputCommand</td>
</tr>
<tr>
<td></td>
<td>The output command contains an invalid value, e.g. the color type is PEXColorTypeIndexed, and the color index value exceeds 65534.</td>
</tr>
<tr>
<td></td>
<td>BadPEXRenderer</td>
</tr>
<tr>
<td></td>
<td>The specified renderer resource identifier is invalid.</td>
</tr>
<tr>
<td></td>
<td>BadPEXStructure</td>
</tr>
<tr>
<td></td>
<td>The specified structure resource identifier is invalid.</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>PEXSetIndividualASF(3)</td>
</tr>
<tr>
<td></td>
<td>PEXCreateLookupTable(3)</td>
</tr>
</tbody>
</table>
NAME  PEXSetTextColorIndex - Set Text Color Index

SYNTAX  void PEXSetTextColorIndex(Display ∗display, XID resource_id, PEXOCRequestType
req_type, unsigned int index)

PARAMETERS  

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>display</td>
<td>A pointer to a display structure returned by a successful XOpenDisplay call.</td>
</tr>
<tr>
<td>resource_id</td>
<td>The resource identifier of the renderer or structure.</td>
</tr>
<tr>
<td>req_type</td>
<td>The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).</td>
</tr>
<tr>
<td>index</td>
<td>The color table index for text.</td>
</tr>
</tbody>
</table>

RETURNS  None

DESCRIPTION  This function creates an output primitive attribute which sets the text color attribute to an indexed color with the value indicated by index. If the specified color index is not defined, color index one is used. If color index one is not defined, the result is implementation-dependent.

ERRORS  

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BadPEXOutputCommand</td>
<td>The color index value exceeds 65534.</td>
</tr>
<tr>
<td>BadPEXRenderer</td>
<td>The specified renderer resource identifier is invalid.</td>
</tr>
<tr>
<td>BadPEXStructure</td>
<td>The specified structure resource identifier is invalid.</td>
</tr>
</tbody>
</table>

SEE ALSO  PEXSetIndividualASF(3)
PEXCreateLookupTable(3)
NAME
PEXSetTextFontIndex - Set Text Font Index

SYNTAX
void PEXSetTextFontIndex(Display ∗display, XID resource_id, PEXOCRequestType
req_type, unsigned int index)

PARAMETERS
display
A pointer to a display structure returned by a successful XOpenDisplay
call.
resource_id
The resource identifier of the renderer or structure.
req_type
The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle
or PEXOCStoreSingle).
index
The text font table index.

RETURNS
None

DESCRIPTION
This function creates an output primitive attribute which sets the text font index attribute.
Index specifies the entry in the current text font lookup table used to render text primitives. If index
is undefined, the default index one is used. If index one is undefined, the result is implementation-dependent.
A text font table index less than 1 will produce a BadPEXOutputCommand error.

ERRORS
BadPEXOutputCommand
The output command contains an invalid value.
BadPEXRenderer
The specified renderer resource identifier is invalid.
BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO
PEXSetIndividualASF(3)
PEXCreateLookupTable(3)

modified November 1995
NAME
PEXSetTextPath - Set Text Path

SYNTAX
void PEXSetTextPath(Display *display, XID resource_id, PEXOCRequestType req_type, int path)

PARAMETERS
- display: A pointer to a display structure returned by a successful XOpenDisplay call.
- resource_id: The resource identifier of the renderer or structure.
- req_type: The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- path: The text (drawing) path (PEXPathRight, PEXPathLeft, PEXPathUp, PEXPathDown).

RETURNS
None

DESCRIPTION
This function creates an output primitive attribute which sets the text path attribute.

ERRORS
- BadPEXOutputCommand: The output command contains an invalid value.
- BadPEXRenderer: The specified renderer resource identifier is invalid.
- BadPEXStructure: The specified structure resource identifier is invalid.
**NAME**
PEXSetTextPrecision - Set Text Precision

**SYNTAX**
PEXSetTextPrecision(Display *display, XID resource_id, PEXOCRequestType req_type, int precision)

**PARAMETERS**
- *display*: A pointer to a display structure returned by a successful XOpenDisplay call.
- *resource_id*: The resource identifier of the renderer or structure.
- *req_type*: The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- *precision*: The text precision (PEXStringPrecision, PEXCharPrecision, PEXStrokePrecision).

**RETURNS**
None

**DESCRIPTION**
This function creates an output primitive attribute which sets the text precision attribute. When text or annotation text is interpreted, the fragments in the text string are rendered in the same text precision. If the font group selected by the current text font index consists of both X and PEX fonts, and if some of the string fragments in the string are rendered in X fonts, the text precision of the entire string must be dropped to at least PEXCharPrecision.

If a character set value is not available in the current font group, then the entire string is rendered using the default font group. If a character set value is not available in the default font group, then that portion of the string is rendered in an implementation-dependent manner.

**ERRORS**
- BadPEXOutputCommand: The output command contains an invalid value.
- BadPEXRenderer: The specified renderer resource identifier is invalid.
- BadPEXStructure: The specified structure resource identifier is invalid.

**SEE ALSO**
PEXSetIndividualASF(3)
NAME
PEXSetViewIndex - Set View Index

SYNTAX
void PEXSetViewIndex(Display *display, XID resource_id, PEXOCRequestType req_type,
unsigned int index)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
index The view table index.

RETURNS
None

DESCRIPTION
This function creates an output primitive attribute which sets the view index attribute. If the specified view index is not defined, the default index zero is used. If view index zero is undefined, the default values are used: all clip flags are on; the clip limits are set to <0,0,0>,<1,1,1>; orientation and mapping are the identity matrices.

ERRORS
BadPEXOutputCommand
The view index value exceeds 65534.
BadPEXRenderer
The specified renderer resource identifier is invalid.
BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO
PEXCreateLookupTable(3)
**NAME**
PEXSetWorkstationBufferMode - Set Workstation Buffer Mode

**SYNTAX**
```c
void PEXSetWorkstationBufferMode(Display *display, PEXWorkstation workstation, int buffer_mode)
```

**PARAMETERS**
- `display` A pointer to a display structure returned by a successful XOpenDisplay call.
- `workstation` The resource identifier of the workstation.
- `buffer_mode` The workstation buffering mode (PEXSingleBuffer or PEXDoubleBuffer).

**RETURNS**
None

**DESCRIPTION**
This function sets the requested buffer mode of the specified workstation. If the workstation's display surface attribute is PEXEmpty, or if the dynamic modification for buffer mode is PEXIMM, the current buffer mode is set to the specified buffer mode and the buffer update is set to PEXNotPending; otherwise, the buffer update is set to PEXPending and the current buffer mode is not changed.

Buffer mode may be one of the following values: PEXSingleBuffer or PEXDoubleBuffer. An error will be generated if the buffer mode is PEXDoubleBuffer and the server cannot allocated the second image buffer.

**DATA STRUCTURES**
```c
typedef XID PEXWorkstation;
```

**ERRORS**
- **BadAlloc**
  The server failed to allocate resources needed for double-buffering.
- **BadPEXWorkstation**
  The specified workstation resource identifier is invalid.
- **BadValue**
  The specified buffer mode is invalid.

**SEE ALSO**
PEXSetWorkstationDisplayUpdateMode(3)
PEXGetImpDepConstants(3)
### NAME
PEXSetWorkstationDisplayUpdateMode - Set Workstation Display Update Mode

### SYNTAX
```c
void PEXSetWorkstationDisplayUpdateMode(Display *display, PEXWorkstation workstation, int update_mode);
```

### PARAMETERS
- **display**
  A pointer to a display structure returned by a successful XOpenDisplay call.
- **workstation**
  The resource identifier of the workstation.
- **update_mode**
  The workstation display update mode (`PEXVisualizeEach`, `PEXVisualizeEasy`, `PEXVisualizeNone`, `PEXSimulateSome`, `PEXVisualizeWhenever`).

### RETURNS
None

### DESCRIPTION
This function sets the display update attribute of the specified `workstation`. This attribute defines how changes to the display surface will be visualized. The supported values for display update mode are inquirable via `PEXGetEnumTypeInfo(3)`.

If double-buffering is enabled, the display update mode affects which buffer is rendered into during traversal. If the display update mode is `PEXVisualizeEach`, `PEXVisualizeWhenever` or `PEXVisualizeNone`, output primitives are rendered into the back (undisplayed) buffer while the structure network is being traversed. When the traversal is complete, the front and back buffers are swapped, so the rendered image is displayed. If the display update mode is `PEXVisualizeEasy` or `PEXSimulateSome`, output primitives are always rendered into the front (displayed) buffer.

### DATA STRUCTURES
```c
typedef XID PEXWorkstation;
```

### ERRORS
- **BadPEXWorkstation**
  The specified workstation resource identifier is invalid.
- **BadValue**
  The specified display update mode is invalid.

### SEE ALSO
- `PEXSetWorkstationBufferMode(3)`
- `PEXGetEnumTypeInfo(3)`

---

modified November 1995 381
NAME
PEXSetWorkstationHLHSRMode - Set Workstation HLHSR Mode

SYNTAX
void PEXSetWorkstationHLHSRMode(Display *display, PEXWorkstation workstation, int hlhsr_mode)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
workstation The resource identifier of the workstation.
hlhsr_mode The workstation HLHSR mode (PEXHLHSROff, PEXHLHSRZBuffer, PEXHLHSRPainters, PEXHLHSRScanline, PEXHLHSRHidden-LineOnly, PEXHLHSRZBufferID).

RETURNS
None

DESCRIPTION
This function sets the requested HLHSR mode of the specified workstation. If the workstation’s display surface attribute is PEXEmpty, or if the dynamic modification for HLHSR mode is PEXIMM, the current HLHSR mode is set to the specified HLHSR mode and the HLHSR update is set to PEXNotPending; otherwise, the HLHSR update is set to PEXPending and the current HLHSR mode is not changed.

DATA STRUCTURES
typedef XID PEXWorkstation;

ERRORS
BadAlloc
The server failed to allocate resources needed for HLHSR.
BadPEXWorkstation
The specified workstation resource identifier is invalid.
BadValue
The specified HLHSR mode is invalid.
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXSetWorkstationViewPriority - Set Workstation View Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td>void PEXSetWorkstationViewPriority(Display *display, PEXWorkstation workstation, unsigned int index1, unsigned int index2, int priority)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>display</td>
</tr>
<tr>
<td></td>
<td>workstation</td>
</tr>
<tr>
<td></td>
<td>index1</td>
</tr>
<tr>
<td></td>
<td>index2</td>
</tr>
<tr>
<td></td>
<td>priority</td>
</tr>
<tr>
<td>RETURNS</td>
<td>None</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>This function sets the relative priorities of entries in the workstation’s current view table. The priority of the first view table entry with respect to the second view table entry is set to the next higher or lower priority as indicated by the specified priority. These priorities are used to determine the order in which view table entries are tested when selecting the inverse viewing transformation to use for transforming from device coordinates to world coordinates.</td>
</tr>
<tr>
<td>DATA STRUCTURES</td>
<td>typedef XID PEXWorkstation;</td>
</tr>
<tr>
<td>ERRORS</td>
<td>BadPEXWorkstation</td>
</tr>
<tr>
<td></td>
<td>BadValue</td>
</tr>
</tbody>
</table>
## NAME
PEXSetWorkstationViewRep - Set Workstation View Representation

## SYNTAX
```c
void PEXSetWorkstationViewRep(Display *display, PEXWorkstation workstation,
                              unsigned int view_index, PEXViewEntry *values)
```

## PARAMETERS
- **display**: A pointer to a display structure returned by a successful `XOpenDisplay` call.
- **workstation**: The resource identifier of the workstation.
- **view_index**: The view index.
- **values**: A pointer to the view representation values.

## RETURNS
None

## DESCRIPTION
This function sets the specified view table entry of the requested view table in the specified workstation. If the dynamic modification for the view table is `PEXIMM`, the current view table entry is set to the specified view representation and the view update is set to `PEXNotPending`; otherwise, the view update is set to `PEXPending` and the current view table is not changed.

## DATA STRUCTURES
```c
typedef XID PEXWorkstation;

typedef struct {
    unsigned short clip_flags;
    unsigned short reserved;
    PEXNPCSubVolume clip_limits;
    PEXMatrix orientation;
    PEXMatrix mapping;
} PEXViewEntry;

typedef struct {
    PEXCoord min;
    PEXCoord max;
} PEXNPCSubVolume;

typedef struct {
    float x;
    float y;
    float z;
} PEXCoord;

typedef float PEXMatrix[4][4];
```
<table>
<thead>
<tr>
<th>ERRORS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BadAlloc</td>
<td>The view table is full.</td>
</tr>
<tr>
<td>BadPEXWorkstation</td>
<td>The specified workstation resource identifier is invalid.</td>
</tr>
</tbody>
</table>
PEXSetWorkstationViewport - Set Workstation Viewport

**SYNTAX**

```c
void PEXSetWorkstationViewport(Display *display, PEXWorkstation workstation, PEXViewport *viewport)
```

**PARAMETERS**

- `display`: A pointer to a display structure returned by a successful `XOpenDisplay` call.
- `workstation`: The resource identifier of the workstation.
- `viewport`: The workstation viewport.

**RETURNS**

None

**DESCRIPTION**

This function sets the requested workstation viewport of the specified workstation. If the dynamic modification for workstation viewport is `PEXIMM`, the current workstation viewport is set to the specified viewport and the workstation update is set to `PEXNotPending`; otherwise, the workstation update is set to `PEXPending` and the current workstation viewport is not changed.

**DATA STRUCTURES**

```c
typedef XID PEXWorkstation;

typedef struct {
    PEXDeviceCoord min;
    PEXDeviceCoord max;
    PEXSwitch use_drawable;
    unsigned char reserved[3];
} PEXViewport;

typedef struct {
    short x;
    short y;
    float z;
} PEXDeviceCoord;

typedef unsigned char PEXSwitch;
```

**ERRORS**

- **BadPEXWorkstation**
  The specified workstation resource identifier is invalid.
- **BadValue**
  The specified value for use_drawable is invalid.
NAME  PEXSetWorkstationWindow - Set Workstation Window

SYNTAX  void PEXSetWorkstationWindow(Display *display, PEXWorkstation workstation, PEXNPCSubVolume *workstation_window)

PARAMETERS  display  A pointer to a display structure returned by a successful XOpenDisplay call.
workstation  The resource identifier of the workstation.
workstation_window  The workstation window.

RETURNS  None

DESCRIPTION  This function sets the requested NPC subvolume of the specified workstation. If the dynamic modification for the workstation window is PEXIMM, the current workstation window is set to the specified NPC subvolume and the workstation update is set to PEX-NotPending; otherwise, the workstation update is set to PEXPending and the current workstation window is not changed.

DATA STRUCTURES

typedef XID PEXWorkstation;
typedef struct {
    PEXCoord min;
    PEXCoord max;
} PEXNPCSubVolume;
typedef struct {
    float x;
    float y;
    float z;
} PEXCoord;

ERRORS  BadPEXWorkstation
The specified workstation resource identifier is invalid.
**NAME**
PEXStartOCs - Start Encoded Output Commands

**SYNTAX**
Status PEXStartOCs(Display ∗display, XID resource_id, PEXOCRequestType req_type, int float_format, int oc_count, int word_count)

**PARAMETERS**
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
float_format The floating point format of the output command data (PEXIEEE_754_32, PEXDEC_F_Floating, PEXIEEE_754_64, PEXDEC_D_Floating).
oc_count The number of output commands to be sent.
word_count The number of four-byte words of data for the total size of the output commands.

**RETURNS**
Zero is unsuccessful, non-zero otherwise.

**DESCRIPTION**
This function locks the display. Only PEXCopyBytesToOC or PEXGetOCAddr may be called between pairs of PEXStartOCs and PEXFinishOCs. Do not call anything else that may lock the display as this will result in deadlock.

The first output command is guaranteed to start on a four-byte boundary. Output command data may be copied into the transport buffer by calling PEXCopyBytesToOC. Output command data may be written directly by the application by calling PEXGetOCAddr to get a pointer to memory in the transport buffer.

PEXFinishOCs must be called after all the data has been specified.

The application is responsible for writing valid protocol and the correct number of words requested.

If the requested number of words is too large for the display connection (each server has a maximum request size), the function will return unsuccessfully. If this occurs, and the number of output commands was greater than one, the application should try specifying the data for a single output command at a time. If the size of a single output command is too large for the display connection, the function will return unsuccessfully.

**ERRORS**
BadPEXFloatingPointFormat
The specified floating point format is invalid or unsupported.

BadPEXOutputCommand
The output command contains an invalid value.

BadPEXRenderer
The specified renderer resource identifier is invalid.
BadPEXStructure

The specified structure resource identifier is invalid.

SEE ALSO

- PEXFinishOCs(3)
- PEXCopyBytesToOC(3)
- PEXGetOCAddr(3)
**NAME**
PEXText - 3D Text Primitive

**SYNTAX**
```c
void PEXText(Display *display, XID resource_id, PEXOCRequestType req_type, PEXCoord *origin, PEXVector *vector1, PEXVector *vector2, int length, char *string)
```

**PARAMETERS**
- `display` A pointer to a display structure returned by a successful `XOpenDisplay` call.
- `resource_id` The resource identifier of the renderer or structure.
- `req_type` The request type for the output command (`PEXOCRender`, `PEXOCStore`, `PEXOCRenderSingle` or `PEXOCStoreSingle`).
- `origin` The origin of the text string.
- `vector1` A vector defining the positive x-direction of the text local coordinate system.
- `vector2` A vector defining the positive y-direction of the text local coordinate system.
- `length` The number of bytes in the text string.
- `string` A pointer to the text string.

**RETURNS**
None

**DESCRIPTION**
This function creates a text output primitive. The first character set in the text font will be used.

The text string is located on a plane defined by its position and direction vectors. The origin defines the position, in model coordinates, at which to render the text string. The two direction vectors define the positive x- and y-directions of the text local coordinate system. If the two vectors are parallel or if one of the vectors has zero length, the vector values `<1,0,0>` and `<0,1,0>` are used.

During rendering, the string’s position is transformed to a position in device coordinates. The string’s color is only affected by depth-cueing and is mapped to a device color. The text string is clipped depending on the current text precision attribute. If the text precision is `PEXStringPrecision`, clipping is done in an implementation-dependent fashion. If the text precision is `PEXCharPrecision`, clipping is done on at least a character-by-character basis. If the text precision is `PEXStrokePrecision`, clipping is performed at the clipping boundaries for each character.

Depending on the text ASF values, the text color, text precision, character expansion, character spacing, and text font attributes are obtained either directly from the current text attribute values or from the entry in the text bundle specified by the current text bundle index attribute. The current character height, text path, text alignment attributes and character up vector are also used to render the text string. The directions specified by the character up vector and text path are relative to the text local coordinate system.
<table>
<thead>
<tr>
<th>DATA STRUCTURES</th>
<th>See PEXlib.h.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERRORS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BadPEXRenderer</td>
</tr>
<tr>
<td></td>
<td>The specified renderer resource identifier is invalid.</td>
</tr>
<tr>
<td></td>
<td>BadPEXStructure</td>
</tr>
<tr>
<td></td>
<td>The specified structure resource identifier is invalid.</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>PEXSetTextFontIndex(3)</td>
</tr>
<tr>
<td></td>
<td>PEXSetTextPrecision(3)</td>
</tr>
<tr>
<td></td>
<td>PEXSetCharExpansion(3)</td>
</tr>
<tr>
<td></td>
<td>PEXSetCharSpacing(3)</td>
</tr>
<tr>
<td></td>
<td>PEXSetTextColorIndex(3)</td>
</tr>
<tr>
<td></td>
<td>PEXSetTextColor(3)</td>
</tr>
<tr>
<td></td>
<td>PEXSetCharHeight(3)</td>
</tr>
<tr>
<td></td>
<td>PEXSetCharUpVector(3)</td>
</tr>
<tr>
<td></td>
<td>PEXSetTextPath(3)</td>
</tr>
<tr>
<td></td>
<td>PEXSetTextAlignment(3)</td>
</tr>
<tr>
<td></td>
<td>PEXSetTextBundleIndex(3)</td>
</tr>
</tbody>
</table>
### NAME
PEXText2D - 2D Text Primitive

### SYNTAX
```c
void PEXText2D(Display ∗display, XID resource_id, PEXOCRequestType req_type,
    PEXCoord2D ∗origin, int length, char ∗string)
```

### PARAMETERS
- **display**: A pointer to a display structure returned by a successful `XOpenDisplay` call.
- **resource_id**: The resource identifier of the renderer or structure.
- **req_type**: The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
- **origin**: The origin of the text string.
- **length**: The number of bytes in the text string.
- **string**: A pointer to the text string.

### RETURNS
None

### DESCRIPTION
This function creates a 2D text output primitive.

This function is like PEXText(3), except that the origin position consists of only x- and y-components. The z-component is assumed to be zero. This primitive is two-dimensional only in that the z-component is implied. Geometry transformations are still carried out in three dimensions.

### DATA STRUCTURES
See PEXlib.h.

### ERRORS
- **BadPEXRenderer**: The specified renderer resource identifier is invalid.
- **BadPEXStructure**: The specified structure resource identifier is invalid.

### SEE ALSO
- PEXSetTextFontIndex(3)
- PEXSetTextPrecision(3)
- PEXSetCharExpansion(3)
- PEXSetCharSpacing(3)
- PEXSetTextColorIndex(3)
- PEXSetTextColor(3)
- PEXSetCharHeight(3)
- PEXSetCharUpVector(3)
- PEXSetTextPath(3)
- PEXSetTextAlignment(3)
- PEXSetTextBundleIndex(3)
NAME
PEXTransformPoints - utility function

SYNTAX
int PEXTransformPoints(PEXMatrix transform, int count, PEXCoord *points, PEXCoord *points_return)

PARAMETERS
transform The transformation matrix to apply to the points.
count The number of points to transform.
points A pointer to an array of 3D points to transform.
points_return A pointer to an array in which to store the transformed points.

RETURNS
Zero if successful; otherwise, one of the following:
PEXBadHomoCoord
One or more of the transformed points has a homogeneous coordinate of 0.

DESCRIPTION
This function applies the specified homogeneous transformation matrix to the list of points. In applying the transformation, the points are first converted to homogeneous points by assigning them a homogeneous coordinate of 1. The transformation is then applied:

\[ P' = T \cdot P \]

Where \( P \) is the point, treated as a column vector, and \( T \) is the transformation matrix. The points are then mapped to 3D by dividing their first three coordinates by the computed homogeneous coordinate.

If the function returns unsuccessfullly, all points other than those with a homogeneous coordinate of 0 will be transformed and returned.

If the return array is the same as the input array, the function will overwrite the input values with the transformed values.

ERRORS
None

SEE ALSO
PEXTransformPoints2D(3)
PEXTransformPoints4D(3)
PEXTransformPoints2DH(3)
**NAME**
PEXTransformPoints2D - utility function

**SYNTAX**
```c
int PEXTransformPoints2D(PEXMatrix3x3 transform, int count, PEXCoord2D *points,
                         PEXCoord2D *points_return)
```

**PARAMETERS**
- `transform`: The transformation matrix to apply to the points.
- `count`: The number of points to transform.
- `points`: A pointer to an array of 2D points to transform.
- `points_return`: A pointer to an array in which to store the transformed points.

**RETURNS**
Zero if successful; otherwise, one of the following:
- **PEXBadHomoCoord**: One or more of the transformed points has a homogeneous coordinate of 0.

**DESCRIPTION**
This function applies the specified homogeneous transformation matrix to the list of `points`. In applying the transformation, the `points` are first converted to homogeneous points by assigning them a homogeneous coordinate of 1. The transformation is then applied:

\[ P' = T \cdot P \]

Where \( P \) is the point, treated as a column vector, and \( T \) is the transformation matrix. The `points` are then mapped to 2D by dividing their first three coordinates by the computed homogeneous coordinate.

If the function returns unsuccessfully, all `points` other than those with a homogeneous coordinate of 0 will be transformed and returned.

If the return array is the same as the input array, the function will overwrite the input values with the transformed values.

**ERRORS**
None

**SEE ALSO**
- PEXTransformPoints(3)
- PEXTransformPoints4D(3)
- PEXTransformPoints2DH(3)
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXTransformPoints2DH - utility function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td>void PEXTransformPoints2DH(PEXMatrix3x3 transform, int count, PEXCoord *points, PEXCoord *points_return)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>transform The transformation matrix to apply to the points.</td>
</tr>
<tr>
<td></td>
<td>count The number of points to transform.</td>
</tr>
<tr>
<td></td>
<td>points A pointer to an array of 2D homogeneous points to transform.</td>
</tr>
<tr>
<td></td>
<td>points_return A pointer to an array in which to store the transformed points.</td>
</tr>
<tr>
<td>RETURNS</td>
<td>None</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>This function applies the specified homogeneous transformation matrix to the list of 2D homogeneous points (P = x, y, w). The transformation is applied: P' = T x P</td>
</tr>
<tr>
<td></td>
<td>Where P is the point, treated as a column vector, and T is the transformation matrix. The function returns P' = (x', y', w').</td>
</tr>
<tr>
<td></td>
<td>If the return array is the same as the input array, the function will overwrite the input values with the transformed values.</td>
</tr>
<tr>
<td>ERRORS</td>
<td>None</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>PEXTransformPoints(3)</td>
</tr>
<tr>
<td></td>
<td>PEXTransformPoints2D(3)</td>
</tr>
<tr>
<td></td>
<td>PEXTransformPoints4D(3)</td>
</tr>
</tbody>
</table>
**NAME**  
PEXTransformPoints4D - utility function

**SYNTAX**  
void PEXTransformPoints4D(PEXMatrix transform, int count, PEXCoord4D *points,  
PEXCoord4D *points_return)

**PARAMETERS**  
- *transform*  
The transformation matrix to apply to the points.
- *count*  
The number of points to transform.
- *points*  
A pointer to an array of 4D points to transform.
- *points_return*  
A pointer to an array in which to store the transformed points.

**RETURNS**  
None

**DESCRIPTION**  
This function applies the specified homogeneous transformation matrix to the list of 3D homogeneous points \( P = (x, y, z, w) \). The transformation is applied:

\[
P' = T x P
\]

Where \( P \) is the point, treated as a column vector, and \( T \) is the transformation matrix. The function returns \( P' = (x', y', z', w') \).

If the return array is the same as the input array, the function will overwrite the input values with the transformed values.

**ERRORS**  
None

**SEE ALSO**  
PEXTransformPoints(3)  
PEXTransformPoints2D(3)  
PEXTransformPoints2DH(3)
NAME
PEXTransformVectors - utility function

SYNTAX
void PEXTransformVectors(PEXMatrix transform, int count, PEXVector *vectors, PEXVector *vectors_return)

PARAMETERS
transform The transformation matrix to apply to the vectors.
count The number of vectors to transform.
vectors A pointer to the array of 3D vectors to transform.
vectors_return A pointer to an array in which to store the transformed vectors.

RETURNS
None

DESCRIPTION
This function applies the upper 3x3 submatrix of the specified transformation matrix to the list of 3D vectors. The transformation is applied:

\[ V' = T'xV \]

Where \( V \) is the vector, treated as a column vector, and \( T' \) is the upper 3x3 sub-matrix of \( transform \).

If the return array is the same as the input array, the function will overwrite the input values with the transformed values.

ERRORS
None

SEE ALSO
PEXTransformVectors2D(3)
PEXNormalizeVectors(3)

modified November 1995
NAME PEXTransformVectors2D - utility function

SYNTAX void PEXTransformVectors2D(PEXMatrix3x3 transform, int count, PEXVector2D *vectors, PEXVector2D *vectors_return)

PARAMETERS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>transform</td>
<td>The transformation matrix to apply to the vectors.</td>
</tr>
<tr>
<td>count</td>
<td>The number of vectors to transform.</td>
</tr>
<tr>
<td>vectors</td>
<td>A pointer to the array of 2D vectors to transform.</td>
</tr>
<tr>
<td>vectors_return</td>
<td>A pointer to an array in which to store the transformed vectors.</td>
</tr>
</tbody>
</table>

RETURNS None

DESCRIPTION This function applies the upper 2x2 submatrix of the specified transformation matrix to the list of 2D vectors. The transformation is applied:

\[ V' = T'xV \]

Where \( V \) is the vector, treated as a column vector, and \( T' \) is the upper 2x2 sub-matrix of \( transform \).

If the return array is the same as the input array, the function will overwrite the input values with the transformed values.

ERRORS None

SEE ALSO PEXTransformVectors(3) PEXNormalizeVectors2D(3)
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXTranslate - utility function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td><code>void PEXTranslate(PEXVector *trans_vector, PEXMatrix matrix_return)</code></td>
</tr>
<tr>
<td>PARAMETERS</td>
<td><code>trans_vector</code> Vector containing the X, Y and Z translation factors.</td>
</tr>
<tr>
<td></td>
<td><code>matrix_return</code> Matrix into which rotation matrix is stored.</td>
</tr>
<tr>
<td>RETURNS</td>
<td>None</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>Creates a translation matrix that translates objects by the given translation vector.</td>
</tr>
<tr>
<td>ERRORS</td>
<td>None</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>PEXTranslate2D(3)</td>
</tr>
<tr>
<td>NAME</td>
<td>PEXTranslate2D - utility function</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>SYNTAX</td>
<td>void PEXTranslate2D(PEXVector2D *trans_vector, PEXMatrix3x3 matrix_return)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td></td>
</tr>
<tr>
<td>trans_vector</td>
<td>Vector containing the X, Y and Z translation factors.</td>
</tr>
<tr>
<td>matrix_return</td>
<td>Matrix into which rotation matrix is stored.</td>
</tr>
<tr>
<td>RETURNS</td>
<td>None</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>Creates a 3X3 translation matrix that translates objects by the given translation vector.</td>
</tr>
<tr>
<td>ERRORS</td>
<td>None</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>PEXTranslate(3)</td>
</tr>
</tbody>
</table>
NAME
PEXTriangleStrip - 3D Triangle Strip Primitive

SYNTAX
void PEXTriangleStrip(Display *display, XID resource_id, PEXOCRequestType req_type,
unsigned int facet_attributes, unsigned int vertex_attributes, int color_type, PEXArrayOfFacetData facet_data, unsigned int count, PEXArrayOfVertex vertices)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
resource_id The resource identifier of the renderer or structure.
req_type The request type for the output command (PEXOCRender, PEXOCStore, PEXOCRenderSingle or PEXOCStoreSingle).
facet_attributes A mask indicating the facet attributes provided (PEXGANone, PEXGAColor, PEXGANormal).
vertex_attributes A mask indicating the vertex attributes provided (PEXGANone, PEXGAColor, PEXGANormal).
color_type The type of color data provided (PEXColorTypeIndexed, PEXColorTypeRGB, PEXColorTypeCIE, PEXColorTypeHSV, PEXColorTypeHLS, PEXColorTypeRGB8, PEXColorTypeRGB16).
facet_data An array of facet data.
count The number of vertices.
vertices An array of vertices defining the triangle strip.

RETURNS
None

DESCRIPTION
This function creates a triangle strip output primitive.
The triangle strip is created from the array of vertices. There are two less facets in the strip than the number of vertices. The first triangle in the strip is formed from the first three vertices in the list, the second triangle is formed by the second through the fourth vertices in the list, etc., up to the last triangle, which is formed by the last three vertices in the list. There must be an entry in the facet data array for each facet, if facet data is indicated by the facet attributes.

Normal s are assumed to be unit length vectors. If not unit length, the result is implementation-dependent.

A triangle strip with fewer than three vertices is considered degenerate. It is stored in a structure, but when rendered, the primitive is ignored and has no visual effect.

All other aspects of this primitive are the same as PEXFillAreaWithData(3).

DATA STRUCTURES
See PEXlib.h.

modified November 1995
ERRORS

BadPEXOutputCommand
The output command contains an invalid value.

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO

PEXSetInteriorStyle(3)
PEXSetInteriorStyleIndex(3)
PEXSetSurfaceColorIndex(3)
PEXSetSurfaceColor(3)
PEXSetReflectionAttributes(3)
PEXSetReflectionModel(3)
PEXSetSurfaceInterpMethod(3)
PEXSetBFInteriorStyle(3)
PEXSetBFInteriorStyleIndex(3)
PEXSetBFSurfaceColorIndex(3)
PEXSetBFSurfaceColor(3)
PEXSetBFReflectionAttributes(3)
PEXSetBFReflectionModel(3)
PEXSetBFSurfaceInterpMethod(3)
PEXSetFacetCullingMode(3)
PEXSetFacetDistinguishFlag(3)
PEXSetPatternSize(3)
PEXSetPatternAttributes(3)
PEXSetPatternAttributes2D(3)
PEXSetInteriorBundleIndex(3)
PEXSetSurfaceEdgeFlag(3)
PEXSetSurfaceEdgeType(3)
PEXSetSurfaceEdgeWidth(3)
PEXSetSurfaceEdgeColor(3)
PEXSetSurfaceEdgeColorIndex(3)
PEXSetEdgeBundleIndex(3)
### NAME
PEXUnloadFont - Unload PEX Font

### SYNTAX
```c
void PEXUnloadFont(Display *display, PEXFont font)
```

### PARAMETERS
- **display**
  A pointer to a display structure returned by a successful `XOpenDisplay` call.
- **font**
  The resource identifier of the PEX font.

### RETURNS
None

### DESCRIPTION
This function deletes the association between the resource identifier and the PEX font. The PEX font itself will be freed when no other resource references it.

### DATA STRUCTURES
```c
typedef XID PEXFont;
```

### ERRORS
- **BadPEXFont**
The specified font resource identifier is invalid.

### SEE ALSO
PEXLoadFont(3)

---

modified November 1995
<table>
<thead>
<tr>
<th>NAME</th>
<th>PEXUnpostAllStructures - Unpost All Structures from Workstation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX</td>
<td>void PEXUnpostAllStructures(Display *display, PEXWorkstation workstation)</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>display A pointer to a display structure returned by a successful XOpenDisplay call. workstation The resource identifier of the workstation.</td>
</tr>
<tr>
<td>RETURNS</td>
<td>None</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>This function removes all structures from the specified workstation’s posted structure list.</td>
</tr>
<tr>
<td>DATA STRUCTURES</td>
<td>typedef XID PEXWorkstation;</td>
</tr>
<tr>
<td>ERRORS</td>
<td>BadPEXWorkstation The specified workstation resource identifier is invalid.</td>
</tr>
<tr>
<td>SEE ALSO</td>
<td>PEXUnpostStructure(3)</td>
</tr>
</tbody>
</table>
NAME  
PEXUnpostStructure - Unpost Structure from Workstation

SYNTAX  
void PEXUnpostStructure(Display *display, PEXWorkstation workstation, PEXStructure structure)

PARAMETERS  
display  
A pointer to a display structure returned by a successful XOpenDisplay call.

workstation  
The resource identifier of the workstation.

structure  
The resource identifier of the structure.

RETURNS  
None

DESCRIPTION  
This function removes the specified structure from the workstation's posted structure list. If the structure is not found in the list, an error is generated.

DATA STRUCTURES  
typedef XID PEXWorkstation;
typedef XID PEXStructure;

ERRORS  
BadPEXStructure
The specified structure resource identifier is invalid.

BadPEXWorkstation
The specified workstation resource identifier is invalid.

SEE ALSO  
PEXUnpostAllStructures(3)

modified November 1995
NAME
PEXUpdatePickMeasure - Update Pick Measure

SYNTAX
type PEXPickMeasure;

type PseudoPickRecord {
    PEXPickMeasure volume;
    PEXPickDeviceDCHitBox box;
    PEXPickDataRecord data;
} PEXPickRecord;

type PEXNPCSubVolume PEXPDNPHitVolume;

type PEXCoord {
    min;
    max;
} PEXNPCSubVolume;

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
pick_measure The resource identifier of the pick measure.
pick_device_type The pick device type (PEXPickDeviceDCHitBox or PEXPickDeviceNPCHitVolume).
pick_record A pointer to the pick data record.

RETURNS None

DESCRIPTION
This function updates the specified pick measure resource. If the update causes a primitive to be picked, the pick measure’s pick status will be set to PEXPick and the pick path attribute will be set to the path of the picked primitive. If no primitive was picked, the pick status will be set to PEXNoPick.

The pick path can be used for echoing when the pick measure is created. However, it is not used as a start path from which to start picking.

The input record is a pointer to data used to update the pick measure and depends on the pick device type specified when the pick measure was created. If the pick device type is PEXPickDeviceDCHitBox, the input record should point to a data structure of type PEXPDDCHitBox. If the pick device type is PEXPickDeviceNPCHitVolume, the input record should point to a data structure of type PEXPDNPHitVolume.

DATA STRUCTURES
type XID PEXPickMeasure;

type PseudoPickRecord {
    PEXPickMeasure volume;
    PEXPickDeviceDCHitBox box;
    PEXPickDataRecord data;
} PEXPickRecord;

type PEXNPCSubVolume PEXPDNPHitVolume;

type PEXCoord {
    min;
    max;
} PEXNPCSubVolume;

modified November 1995
typedef struct {
    float x;
    float y;
    float z;
} PEXCoord;

typedef struct {
    PEXDeviceCoord2D position;
    float distance;
} PEXPDDCHitBox;

typedef struct {
    short x;
    short y;
} PEXDeviceCoord2D;

typedef struct 
{
    unsigned short length; /* number of bytes in record */
    char *record;
} PEXPickDataRecord;

ERRORS

BadPEXPath
    The specified path is invalid.

BadPEXPickMeasure
    The specified pick measure resource identifier is invalid.

SEE ALSO
PEXFreePickMeasure(3)
NAME
PEXUpdateWorkstation - Update Workstation

SYNTAX
void PEXUpdateWorkstation(Display *display, PEXWorkstation workstation)

PARAMETERS
display A pointer to a display structure returned by a successful XOpenDisplay call.
workstation The resource identifier of the workstation.

RETURNS
None

DESCRIPTION
This function will perform actions identical to PEXRedrawAllStructures(3) on the specified workstation if the workstation’s visual state is currently set to PEXDeferred or PEXSimulated.

DATA STRUCTURES
typedef XID PEXWorkstation;

ERRORS
BadPEXWorkstation
The specified workstation resource identifier is invalid.

SEE ALSO
PEXRedrawAllStructures(3)
NAME
PEXViewMappingMatrix - utility function

SYNTAX
int PEXViewMappingMatrix(PEXCoord2D *frame, PEXNPCSubVolume *viewport, int perspective, PEXCoord *prp, double view_plane, double back_plane, double front_plane, PEXMatrix matrix_return)

PARAMETERS
frame Array of 2 2D VRC locations which mark a rectangle in the view plane.
viewport NPC viewport into which the frame gets mapped.
perspective Flag to indicate whether a perspective view is desired, a value of True. Requests that perspective be applied.
prp Projection reference point.
view_plane VRC position of view plane w.r.t. the VRP.
back_plane VRC position of the back plane w.r.t. the VRP.
front_plane VRC position of the front plane w.r.t. the VRP.
matrix_return Matrix into which result is stored.

RETURNS
Zero if successful; otherwise, one of the following:
PEXBadLimits
PEXBadViewport
PEXBadPlanes
PEXBadPRP

DESCRIPTION
This function creates a view mapping matrix that transforms a volume specified in view reference coordinates (VRC) to a volume in normalized projection coordinates (NPC). This matrix is used in conjunction with a view orientation matrix as the viewing matrices for a designated view.

The axes of VRC form a right-handed coordinate system. The z axis is along the VPN (view plane normal, see The Y axis is fixed by VUP, and the X axis is determined so that the three axes form a right-handed coordinate system.

The front plane, back plane, and view plane all define planes in VRC parallel to the VRC x-y plane. The location of front_plane and back_plane along the z axis of VRC defines the front and back of the volume of VRC that will be mapped to the specified NPC viewport. The view plane locates the view frame or "window" on the VRC z axis. The two points in frame determine the size of the view window by specifying lower left (frame[0]) and upper right (frame[1]) x and y VRC points of the window on the view plane. These values taken together establish the volume of VRC space that is mapped into the NPC viewport.

The type of projection may be parallel or perspective. The projection reference point (PRP) orients the projectors defining the surfaces of the view volume. If perspective indicator is False, then the projection type is parallel and the projectors are all parallel to the vector joining the projection reference point and the center of the view window (located on the view plane). If perspective is True, then the projectors all converge at the
projection reference point. Thus, the view volume is a parallelepiped for parallel views, and a truncated pyramid for perspective views. When specifying NPC, the X, Y and Z limits must be as follows:

\[ \text{xmin} < \text{xmax}, \text{ymin} < \text{ymax}, \text{zmin} \leq \text{zmax} \]

**DATA STRUCTURES**

typedef struct {
    PEXCoord min;
    PEXCoord max;
} PEXNPCSubVolume;

See PEXlib.h.

**ERRORS**

None

**SEE ALSO**

PEXViewMappingMatrix2D(3)
PEXViewOrientationMatrix(3)
PEXViewOrientationMatrix2D(3)

modified November 1995
**NAME**  PEXViewMappingMatrix2D - utility function

**SYNTAX**  
```c
int PEXViewMappingMatrix2D(PEXCoord2D *frame, PEXCoord2D *viewport,
                         PEXMatrix3x3 matrix_return)
```

**PARAMETERS**  
- `frame`  
  Array of 2 2D VRC locations which mark a rectangle to be mapped into the NPC subvolume.
- `viewport`  
  Array of 2 2D NPC coordinates with z = 0 specifying lower left and upper right of the viewport.
- `matrix_return`  
  Matrix in which result is stored.

**RETURNS**  
Zero if successful; otherwise, one of the following:
- `PEXBadLimits`
- `PEXBadViewport`

**DESCRIPTION**  
This function is a 2D shorthand version of PEXViewMappingMatrix(3) with the following parameters assumed:
- front plane distance = 1
- back plane distance = 0
- view plane distance = 0
- viewport z range = [0,1]
- projection type = parallel
- x-y coordinates of projection reference point = center of view window,
  z coordinate = 1.0.

When specifying NPC, the X, Y and Z limits must be as follows:
- xmin < xmax , ymin < ymax , zmin <= zmax

**ERRORS**  
None

**SEE ALSO**  
PEXViewMappingMatrix(3)  
PEXViewOrientationMatrix(3)  
PEXViewOrientationMatrix2D(3)
NAME
PEXViewOrientationMatrix - utility function

SYNTAX
int PEXViewOrientationMatrix(PEXCoord *vrp, PEXVector *vpn, PEXVector *vup, PEXMatrix matrix_return)

PARAMETERS
vrp View reference point.
vpn View plane normal.
vup View up vector.
matrix_return Matrix into which result is stored.

RETURNS
Zero if successful; otherwise, one of the following:
PEXBadVector
Either vpn or vup is zero length.
PEXBadVectors
vup is parallel to vpn.

DESCRIPTION
This function creates a view orientation matrix that transforms world coordinates (WC) to view reference coordinates (VRC). This matrix is used in conjunction with a view mapping matrix as the viewing matrices for a designated view.
The view reference point (VRP) defines the point in world coordinate space that is to be used as the origin of the view reference coordinate system.
The view plane normal (VPN) is a 3D vector defined in world coordinates relative to the view reference point. This gives the direction of the positive Z axis of VRC.
The view up vector (VUP) is a 3D vector defined in world coordinates relative to the view reference point. The projection of this vector onto the plane through the view reference point and perpendicular to the view plane normal defines the Y axis of VRC.
The X axis of VRC is defined such that the VRC system forms a right-handed coordinate system.

ERRORS
None

SEE ALSO
PEXViewOrientationMatrix2D(3)
PEXViewMappingMatrix(3)
PEXViewMappingMatrix2D(3)

modified November 1995
NAME        PEXViewOrientationMatrix2D - utility function

SYNTAX      int PEXViewOrientationMatrix2D(PEXCoord2D *vrp, PEXVector2D *vup, PEXMatrix3x3
            matrix_return)

PARAMETERS  vrp      View reference point.
vup      View up vector.
matrix_return  Matrix into which result is stored.

RETURNS     Zero if successful; otherwise, one of the following:
            PEXBadVector
            vup is zero length.

DESCRIPTION This function creates a view orientation matrix that transforms world coordinates (WC)
to view reference coordinates (VRC). This matrix is used in conjunction with a view mapping matrix as the viewing matrices for a designated view.
The view reference point (VRP) defines the point in world coordinate space that is to be used as the origin of the view reference coordinate system. The point is in the WC z=0 plane.
The view plane normal (VPN) is in the direction of the positive Z axis of WC.
The view up vector (VUP) is a 2D vector in the WC z=0 plane. It determines the positive Y axis of VRC. It’s defined in world coordinates relative to the view reference point.
The X axis of VRC is defined such that the VRC system forms a right-handed coordinate system.

ERRORS      None

SEE ALSO    PEXViewOrientationMatrix(3)
            PEXViewMappingMatrix(3)
            PEXViewMappingMatrix2D(3)
NAME
PEXXCToNPCTransform - utility function

SYNTAX
int PEXXCToNPCTransform(PEXNPCSubVolume *npc_sub_volume, PEXDeviceCoord *viewport, unsigned int window_height, PEXMatrix transform_return)

PARAMETERS
npc_sub_volume A pointer to an NPC subvolume, typically that of a renderer resource.
viewport An array of two device coordinate points defining a viewport, typically that of a renderer resource. The first point in the array is the lower-left corner of the viewport; the second point is the upper-right.
window_height The height of the drawable.
transform_return The returned transformation.

RETURNS
Zero if successful; otherwise, one of the following:
PEXBadViewport (xmin >= xmax, or ymin >= ymax, or zmin > zmax)
PEXBadSubVolume (xmin >= xmax, or ymin >= ymax, or zmin > zmax)

DESCRIPTION
This function computes a transformation matrix to map a drawable point (XC) to NPC coordinates, using the specified NPC subvolume, DC viewport, and drawable height. The returned transformation matrix first transforms the x and y coordinates of points to device coordinates (DC), leaving the z coordinate unmodified. It then applies the viewport-to-subvolume transformation to all coordinates of the resulting points, producing 3D NPC points.

When specifying NPC and DC, the X, Y and Z limits must be as follows:
    xmin < xmax, ymin < ymax, zmin <= zmax

ERRORS
None

SEE ALSO
PEXXCToNPCTransform2D(3)

modified November 1995
**NAME**  
PEXXCToNPCTransform2D - utility function

**SYNTAX**  
```c
int PEXXCToNPCTransform2D(PEXNPCSubVolume *npc_sub_volume,
                         PEXDeviceCoord2D *viewport, unsigned int window_height,
                         PEXMatrix3x3 *transform_return)
```

**PARAMETERS**
- `npc_sub_volume`: A pointer to an NPC subvolume, typically that of a renderer resource.
- `viewport`: An array of two device coordinate points defining a viewport, typically that of a renderer resource. The first point in the array is the lower-left corner of the viewport; the second point is the upper-right.
- `window_height`: The height of the drawable.
- `transform_return`: The returned transformation.

**RETURNS**  
Zero if successful; otherwise, one of the following:
- **PEXBadViewport**  
  (xmin >= xmax, or ymin >= ymax, or zmin > zmax)
- **PEXBadSubVolume**  
  (xmin >= xmax, or ymin >= ymax, or zmin > zmax)

**DESCRIPTION**  
This function computes the 2D transformation matrix to map a drawable point (XC) to NPC coordinates, using the specified NPC subvolume, DC viewport, and drawable height. The returned transformation matrix first transforms the x and y coordinates of drawable points to device coordinates (DC), then applies the 2D components of the viewport-to-subvolume transformation, producing 2D NPC points.

When specifying NPC and DC, the X, Y and Z limits must be as follows:
- xmin < xmax, ymin < ymax, zmin <= zmax

**ERRORS**  
None

**SEE ALSO**  
PEXXCToNPCTransform(3)
Index

A
Accumulation
PEXAccumulateState, 3
Allocation, 1
Annotation Text
PEXAnnotationText, 5
PEXAnnotationText2D, 7
PEXEncodedAnnoText, 65
PEXEncodedAnnoText2D, 67
PEXSetATextAlignment, 290
PEXSetATextHeight, 291
PEXSetATextPath, 292
PEXSetATextStyle, 293
PEXSetATextUpVector, 294
Application Data
PEXApplicationData, 9

B
Back-Face Surface
PEXSetBFInteriorStyle, 295
PEXSetBFInteriorStyleIndex, 296
PEXSetBFRellectionAttributes, 297
PEXSetBFRellectionModel, 298
PEXSetBFSurfaceColor, 299
PEXSetBFSurfaceColorIndex, 300
PEXSetBFSurfaceInterpMethod, 301

C
Cell Array
PEXCellArray, 20
PEXCellArray2D, 22
PEXExtendedCellArray, 97
Color
PEXSetColorApproxIndex, 306
PEXSetEchoColor, 309
PEXSetTextColor, 374
PEXSetTextColorIndex, 375
compilation, 2
Coordinates
PEXMapDCToWC, 229
PEXMapWCToDC, 231
PEXMapXCToNPC, 233
PEXMapXCToNPC2D, 235
PEXNPCToXCTransform, 243
PEXNPCToXCTransform2D, 244
Copy
PEXCopBytesToOC, 33
PEXCopElements, 34
PEXCopLookupTable, 36
PEXCopNameSet, 37
PEXCopPipelineContext, 38
PEXCopSearchContext, 39
PEXCopStructure, 40
Deallocation, 1

Edge
  PEXSetEdgeBundleIndex, 311

Enumerated Type
  PEXFreeEnumInfo, 119
  PEXGetEnumTypeInfo, 171

Errors, 1

Escapes
  PEXEscape, 79
  PEXEscape-ES_ESCAPE_DBLBUFFER, 81
  PEXEscape-ES_ESCAPE_SWAPBUFFER, 83
  PEXEscape-HP_ESCAPE_DFRONT, 84
  PEXEscape-PEXSunEscIdChangeExtRendAttr, 85
  PEXEscape-PEXSunEscIdDefineMarkerType, 86
  PEXEscape-PEXSunEscIdFlushRenderer, 87
  PEXEscapeWithReply, 80
  PEXEscapeWithReply-
    ES_ESCAPE_SWAPBUFFERCONTENT, 88
  PEXEscapeWithReply-
    PEXSunEscIdGetExtRendAttr, 90
  PEXEscapeWithReply-
    PEXSunEscIdGetExtRendAttrDyn, 92
  PEXEscapeWithReply-
    PEXSunEscIdGetMarkerDescr, 93

Files
  header, 2
  include, 2

Fill Area
  PEXFillArea, 103
  PEXFillArea2D, 106
  PEXFillAreaSet, 108
  PEXFillAreaSet2D, 110
  PEXFillAreaSetWithData, 112
  PEXFillAreaWithData, 115
  PEXGeoNormFillArea, 158
  PEXGeoNormFillAreaSet, 159
  PEXGeoNormSetOfFillAreaSets, 161
  PEXSetOfFillAreaSets, 343

Fonts
  PEXFreeFontInfo, 120
  PEXFreeFontNames, 121
  PEXListFonts, 224
  PEXListFontsWithInfo, 225
  PEXLoadFont, 227
  PEXQueryFont, 273
  PEXSetFontFontIndex, 376
  PEXUnloadFont, 403

Free Storage
  PEXFreePCAttributes, 125
  PEXFreePDAttributes, 126
  PEXFreePMAAttributes, 127
  PEXFreeRendererAttributes, 132
  PEXFreeSCAttributes, 133
  PEXFreeTableEntries, 136
  PEXFreeWorkstationAttributes, 138

GDP
  PEXGDP, 139
  PEXGDP2D, 141

GSE
  PEXGSE, 143
  PEXGSE-ESGseCylinderDivisions, 145
  PEXGSE-ESGseCylinderRadius, 146
  PEXGSE-ESGseSphereDivisions, 147
  PEXGSE-ESGseSphereRadius, 148
  PEXGSE-PEXSunGseIdSetAnnotTextSlantAngle, 149
  PEXGSE-PEXSunGseIdSetEndcap, 150
  PEXGSE-PEXSunGseIdSetHighlightColor, 151
  PEXGSE-PEXSunGseIdSetSilhouetteEdgeFlag, 152
  PEXGSE-PEXSunGseIdSetStrokeAAliasParams, 153
  PEXGSE-PEXSunGseIdSetStrokJoin, 155
  PEXGSE-PEXSunGseIdSetSurfTranspCoef, 156
  PEXGSE-PEXSunGseIdSetTextSlantAngle, 157

Header files, 2
I
Include files, 2
Initialization, 1
Introduction, 1

L
Line
PEXSetLineBundleIndex, 325
PEXSetLineColor, 326
PEXSetLineColorIndex, 327
PEXSetLineType, 328
PEXSetLineWidth, 329

Lookup Table
PEXCreateLookupTable, 42
PEXDeleteTableEntries, 59
PEXFreeLookupTable, 122
PEXGetDefinedIndices, 166
PEXGetPredefinedEntries, 190
PEXGetTableEntries, 203
PEXGetTableEntry, 205
PEXGetTableInfo, 207
PEXSetTableEntries, 370

M
Marker
PEXMarkers, 237, 238
PEXSetMarkerBundleIndex, 333
PEXSetMarkerColor, 334
PEXSetMarkerColorIndex, 335
PEXSetMarkerScale, 336
PEXSetMarkerType, 337

Memory
allocation, 1
deallocation, 1
XFree, 1

Model Clipping
PEXBuildTransform, 18
PEXBuildTransform2D, 19
PEXCellArray, 20
PEXCellArray2D, 22
PEXChangeNameSet, 23
PEXChangePickDevice, 24
PEXChangePipelineContext, 27
PEXChangeRenderer, 28
PEXChangeSearchContext, 30

N
Name Set
PEXAddToNameSet, 4
PEXChangeNameSet, 23
PEXCopyNameSet, 37
PEXCreateNameSet, 44
PEXFreeNameSet, 123
PEXGetNameSet, 181
PEXRemoveFromNameSet, 278

Nurb
PEXNURBCurve, 245
PEXNURBSurface, 247

O
Output Commands
PEXCopyBytesToOC, 33
PEXCountOCs, 41
PEXDecodeOCs, 56
PEXEncodeOCs, 64
PEXFreeOCData, 124
PEXGetOCAddr, 182
PEXGetOCAddrMaxSize, 183
PEXGetSizeOCs, 199
PEXSendOCs, 289
PEXChangeStructureRefs, 32
PEXCopyBytesToOC, 33
PEXCOPYElements, 34
PEXCopyLookupTable, 36
PEXCOPYNameSet, 37
PEXCOPYPipelineContext, 38
PEXCOPYSearchContext, 39
PEXCOPYStructure, 40
PEXCountOCs, 41
PEXCreateLookupTable, 42
PEXCreateNameSet, 44
PEXCreatePickMeasure, 45
PEXCreatePipelineContext, 46
PEXCreateRenderer, 49
PEXCreateSearchContext, 51
PEXCreateStructure, 53
PEXCreateWorkstation, 54
PEXDecodeOCs, 56
PEXDeleteBetweenLabels, 57
PEXDeleteElements, 58
PEXDeleteTableEntries, 59
PEXDeleteToLabel, 60
PEXDestroyStructures, 61
PEXElementSearch, 62
PEXEncodedAnnoText, 65
PEXEncodedAnnoText2D, 67
PEXEncodedText, 69
PEXEncodedText2D, 71
PEXEncodeOCs, 64
PEXEndPickAll, 73
PEXEndPickOne, 75
PEXEndRendering, 77
PEXEndStructure, 78
PEXEscape, 79
PEXEscape-ES_ESCAPE_DBBLBUFFER, 81
PEXEscape-ES_ESCAPE_SWAPBUFFER, 83
PEXEscape-HP_ESCAPE_DFRONT, 84
PEXEscape-PEXSunEscIdChangeExtRendAttr, 85
PEXEscape-PEXSunEscIdDefineMarkerType, 86
PEXEscape-PEXSunEscIdFlushRenderer, 87
PEXEscapeWithReply, 80
PEXEscapeWithReply-ES.Escape_SWAPBUFFERCONTENT, 88
PEXEscapeWithReply-PEXSunEscIdGetExtRendAttr, 90
PEXEscapeWithReply-PEXSunEscIdGetExtRendAttrDyn, 92
PEXEscapeWithReply-PEXSunEscIdGetMarkerDescr, 93
PEXExecuteDeferredActions, 95
PEXExecuteStructure, 96
PEXExtendedCellArray, 97
PEXFetchElements, 99
PEXFetchElementsAndSend, 101
PEXFillArea, 103
PEXFillArea2D, 106
PEXFillAreaSet, 108
PEXFillAreaSet2D, 110
PEXFillAreaSetWithData, 112
PEXFillAreaWithData, 115
PEXFinishOCs, 118
PEXFreeEnumInfo, 119
PEXFreeFontInfo, 120
PEXFreeFontNames, 121
PEXFreePickMeasure, 128
PEXFreePickPaths, 129
PEXFreePipelineContext, 130
PEXFreePMAttributes, 127
PEXFreeRenderer, 131
PEXFreeRendererAttributes, 132
PEXFreeSCAttributes, 133
PEXFreeSearchContext, 134
PEXFreeStructurePaths, 135
PEXFreeTableEntries, 136
PEXFreeWorkstation, 137
PEXFreeWorkstationAttributes, 138
PEXGDP, 139
PEXGDP2D, 141
PEXGeoNormFillArea, 158

Index–4
PEXGeoNormFillAreaSet, 159
PEXGeoNormQuadrilateralMesh, 160
PEXGeoNormSetOfFillAreaSets, 161
PEXGeoNormTriangleStrip, 163
PEXGetAncestors, 164
PEXGetDefinedIndices, 166
PEXGetDescendants, 167
PEXGetElementInfo, 169
PEXGetEnumTypeInfo, 1, 171
PEXGetExtensionInfo, 1, 177
PEXGetImpDepConstants, 1, 178
PEXGetOCAddr, 181
PEXGetOCAddrMaxSize, 183
PEXGetPickDevice, 184
PEXGetPickMeasure, 187
PEXGetPipelineContext, 189
PEXGetPredefinedEntries, 190
PEXGetProtocolFloatFormat, 192
PEXGetRendererAttributes, 193
PEXGetRendererDynamics, 195
PEXGetSearchContext, 197
PEXGetSizeOCs, 199
PEXGetStructureInfo, 200
PEXGetStructuresInNetwork, 202
PEXGetTableEntries, 203
PEXGetTableEntry, 205
PEXGetTableInfo, 207
PEXGetWorkstationAttributes, 209
PEXGetWorkstationDynamics, 212
PEXGetWorkstationPostings, 214
PEXGetWorkstationViewRep, 215
PEXGSE, 143
PEXGSE-ESGseCylinderDivisions, 145
PEXGSE-ESGseCylinderRadius, 146
PEXGSE-ESGseSphereDivisions, 147
PEXGSE-ESGseSphereRadius, 148
PEXGSE-PEXSunGseIdSetTextSlantAngle, 155
PEXGSE-PEXSunGseIdSetStrokeJoin, 155
PEXGSE-PEXSunGseIdSetSurfTranspCoef, 156
PEXIdentityMatrix, 217, 218
PEXInvertMatrix, 221
PEXInvertMatrix2D, 222
PEXLabel, 223
PEXlib initialization, 1
PEXlib termination, 1
PEXListFonts, 224
PEXListFontsWithInfo, 225
PEXLoadFont, 227
PEXLookAtViewMatrix, 228
PEXMatrixMult, 241
PEXMatrixMult2D, 242
PEXNormalizeVectors, 252
PEXNormalizeVectors2D, 253
PEXNPCToXCTransform, 243
PEXNPCToXCTransform2D, 244
PEXOrthoProjMatrix, 245
PEXPerspProjMatrix, 246
PEXPickAll, 257
PEXPickOne, 260
PEXPolarViewMatrix, 263
PEXPolyline, 264
PEXPolyline2D, 265
PEXPolylineSetWithData, 266
PEXPolylineSetWithIndices, 268
PEXPostStructure, 268
PEXQuadrilateralMesh, 269
PEXQueryEncodedTextExtents, 271
PEXQueryFont, 273
PEXQueryTextExtents, 274
PEXQueryTextExtents, 274
PEXRedrawAllStructures, 276
PEXRedrawClipRegion, 277
PEXRemoveFromNameSet, 278
PEXRenderElements, 279
PEXRenderNetwork, 281
PEXRestoreModelClipVolume, 282
PEXRotate, 283
PEXRotate2D, 284
PEXRotateGeneral, 285
PEXScale, 286
PEXScale2D, 287
PEXSearchNetwork, 288
PEXSendOCs, 289
PEXSetTextAlignment, 290
PEXSetTextHeight, 291
PEXSetTextPath, 292
PEXSetTextStyle, 293
PEXSetTextUpVector, 294
PEXSetBFInteriorStyle, 295
PEXSetBFInteriorStyleIndex, 296
PEXSetBFRotationAttributes, 297
PEXSetBFRotationModel, 298
PEXSetBFSurfaceColor, 299
PEXSetBFSurfaceColorIndex, 300
PEXSetBFSurfaceInterpMethod, 301
PEXSetCharExpansion, 302
PEXSetCharHeight, 303
PEXSetCharSpacing, 304
PEXSetCharUpVector, 305
PEXSetColorApproxIndex, 306
PEXSetCurveApprox, 307
PEXSetDepthCueIndex, 308
PEXSetEchoColor, 309
PEXSetEdgeBundleIndex, 311
PEXSetEditingMode, 312
PEXSetElementPtr, 313
PEXSetElementPtrAtLabel, 314
PEXSetFacetCullingMode, 315
PEXSetFacetDistinguishFlag, 316
PEXSetGlobalTransform, 317
PEXSetGlobalTransform2D, 318
PEXSetHLHSRID, 319
PEXSetIndividualASF, 320
PEXSetInteriorBundleIndex, 321
PEXSetInteriorStyle, 322
PEXSetInteriorStyleIndex, 323
PEXSetLightSourceState, 324
PEXSetLineWidth, 325
PEXSetLineStyle, 326
PEXSetLineStyleIndex, 327
PEXSetLineType, 328
PEXSetLocalTransform, 330
PEXSetLocalTransform2D, 331
PEXSetMarkerBundleIndex, 333
PEXSetMarkerColor, 334
PEXSetMarkerColorIndex, 335
PEXSetMarkerScale, 336
PEXSetMarkerType, 337
PEXSetModelClipFlag, 338
PEXSetModelClipVolume, 339
PEXSetModelClipVolume2D, 341
PEXSetOfFillAreaSets, 343
PEXSetParaSurfCharacteristics, 351
PEXSetPatternAttributes, 353
PEXSetPatternAttributes2D, 354
PEXSetPatternSize, 355
PEXSetPCAttributeMask, 346
PEXSetPCAttributeMaskAll, 348
PEXSetPickID, 356
PEXSetPolylineInterpMethod, 357
PEXSetPWAttributeMask, 349
PEXSetPWAttributeMaskAll, 350
PEXSetReflectionAttributes, 358
PEXSetReflectionModel, 359
PEXSetRenderingColorModel, 360
PEXSetSurfaceApprox, 361
PEXSetSurfaceColor, 362
PEXSetSurfaceColorIndex, 363
PEXSetSurfaceEdgeColor, 364
PEXSetSurfaceEdgeColorIndex, 365
PEXSetSurfaceEdgeFlag, 366
PEXSetSurfaceEdgeType, 367
PEXSetSurfaceEdgeWidth, 368
Pick Measure, continued
  PEXFreePickMeasure, 128
  PEXGetPickMeasure, 187

Picking
  PEXBeginPickAll, 10
  PEXBeginPickOne, 13
  PEXChangePickDevice, 24
  PEXEndPickAll, 73
  PEXEndPickOne, 75
  PEXFreePickPaths, 129
  PEXGetPickDevice, 184
  PEPickAll, 257
  PEPickOne, 260
  PEXSetPickID, 356

Pipeline Context
  PEXChangePipelineContext, 27
  PEXCopyPipelineContext, 38
  PEXCreatePipelineContext, 46
  PEXFreePipelineContext, 130
  PEXGetPipelineContext, 189

Polyline
  PEXPolyline, 264
  PEXPolyline2D, 265
  PEXPolylineSetWithData, 266

R
Rendering
  PEXBeginRendering, 16
  PEXBeginStructure, 17
  PEXChangeRenderer, 28
  PEXEndRendering, 77
  PEXEndStructure, 78
  PEXFreeRenderer, 131
  PEXGetRendererAttributes, 193
  PEXGetRendererDynamics, 195
  PEXMatchRenderingTargets, 239
  PEXRenderElements, 279
  PEXRenderNetwork, 281
  PEXSetRenderingColorModel, 360

Return values, 1

S
Search Context
  PEXChangeSearchContext, 30
  PEXCopySearchContext, 39
Search Context, continued
  PEXCreateSearchContext, 51
  PEXFreeSearchContext, 134
  PEXGetSearchContext, 197
  PEXSearchNetwork, 288

Structure Elements
  PEXChangeStructureRefs, 32
  PEXCopyElements, 34
  PEXDeleteElements, 58
  PEXFetchElements, 99
  PEXFetchElementsAndSend, 101
  PSEXsetElementPtr, 313
  PSEXsetElementPtrAtLabel, 314

Surface
  PSEXsetSurfaceApprox, 361
  PSEXsetSurfaceColor, 362
  PSEXsetSurfaceColorIndex, 363
  PSEXsetSurfaceEdgeColor, 364
  PSEXsetSurfaceEdgeColorIndex, 365
  PSEXsetSurfaceEdgeFlag, 366
  PSEXsetSurfaceEdgeType, 367
  PSEXsetSurfaceEdgeWidth, 368
  PSEXsetSurfaceInterpMethod, 369

  T
Termination, 1

Text Primitive
  PSEXencodedText, 69
  PSEXencodedText2D, 71

Transport mechanisms, 1

  U
Utility Functions
  PSEXbuildTransform, 18
  PSEXbuildTransform2D, 19
  PSEXGeoNormFillArea, 158
  PSEXGeoNormFillAreaSet, 159
  PSEXGeoNormQuadrilateralMesh, 160
  PSEXGeoNormSetOfFillAreaSets, 161
  PSEXGeoNormTriangleStrip, 163
  PSEXIDentityMatrix, 217, 218
  PSEXInvertMatrix, 221
  PSEXInvertMatrix2D, 222
  PSEXLookAtViewMatrix, 228
  PSEXMapXCToNPC, 233

Utility Functions, continued
  PSEXMapXCToNPC2D, 235
  PSEXMatrixMult, 241
  PSEXMatrixMult2D, 242
  PSEXNormalizeVectors, 252
  PSEXNormalizeVectors2D, 253
  PSEXNPCToXCTransform, 243
  PSEXNPCToXCTransform2D, 244
  PSEXOrthoProjMatrix, 254
  PSEXPerspProjMatrix, 255
  PSEXPolynomialViewMatrix, 263
  PSEXRotate, 283
  PSEXRotate2D, 284
  PSEXRotateGeneral, 285
  PSEXScale, 286
  PSEXScale2D, 287
  PSEXTransformPoints, 393
  PSEXTransformPoints2D, 394
  PSEXTransformPoints2DH, 395
  PSEXTransformPoints4D, 396
  PSEXTransformVectors, 397
  PSEXTransformVectors2D, 398
  PSEXTranslate, 399
  PSEXTranslate2D, 400
  PSEXViewMappingMatrix, 409
  PSEXViewMappingMatrix2D, 411
  PSEXViewOrientationMatrix, 412
  PSEXViewOrientationMatrix2D, 413
  PSEXXCToNPC, 414
  PSEXXCToNPC2D, 415

  W
Workstation
  PSEXCreateWorkstation, 54
  PSEXExecuteDeferredActions, 95
  PSEXFreeWorkstation, 137
  PSEXFreeWorkstationAttributes, 138
  PSEXGetWorkstationAttributes, 209
  PSEXGetWorkstationDynamics, 212
  PSEXGetWorkstationPostings, 214
  PSEXGetWorkstationViewRep, 215
  PSEXSetWorkstationBufferMode, 380
  PSEXSetWorkstationDisplayUpdateMode, 381
  PSEXSetWorkstationHLHSMRMode, 382
  PSEXSetWorkstationViewport, 386

Index–8
Workstation, continued
  PEXSetWorkstationViewPriority, 383
  PEXSetWorkstationViewRep, 384
  PEXSetWorkstationWindow, 387
  PEXUpdateWorkstation, 408

X
XNextEvent, 1
XSync, 1
XSynchronize, 1