Common Graphics Environment (CGE) Extensions 5

This chapter contains all of the reference pages for the Sun supported CGE Extensions. Each utility function can be identified by its name in the NAME section at the top of each reference page.

Extended Change Pipeline Context

NAME:

PEXExtChangePipelineContext - Extended Change Pipeline Context

SYNTAX:

void PEXExtChangePipelineContext (Display **display*, PEXPipelineContext *context*, unsigned long **value_mask*, PEXExtPCAttributes **values*)

PARAMETERS:

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

RETURNS:

None

DESCRIPTION:

This function changes components of the specified extended pipeline context to the values specified. The value mask indicates which attribute values are specified. `PEXSetPCAttributeMaskAll` can be called to set up the non-extended portion of the value mask and `PEXExtSetPCAttributeMask` can be called to set up any portion of the value mask.

DATA STRUCTURES:

```
typedef XID        PEXPipelineContext;
```

See also `PEXlib.h` and `PEXExtlib.h`

ERRORS:

BadPEXColorType
The specified color type is invalid or unsupported.

BadPEXFloatingPointFormat
The specified floating point format is invalid or unsupported.

BadPEXNameSet
The specified name set resource identified 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`, `PEXExtGetPipelineContext`, `PEXExtSetPCAttributeMask`, `PEXExtSetPCAttributeMaskAll`

Extended Change Renderer Attributes

NAME:

PEXExtChangeRenderer - Extended Change Renderer Attributes

SYNTAX:

void PEXExtChangeRenderer(Display **display*, PEXRenderer *renderer*,
unsigned long **value_mask*, PEXExtRendererAttributes **values*)

PARAMETERS:

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

renderer — The resource identifier of the extended renderer.

value_mask — A pointer to an array of two unsigned long.

values — A pointer to new values for attributes in the extended renderer.

RETURNS:

None

DESCRIPTION:

This function changes attributes of the specified extended renderer to the values specified. The value mask indicates which attribute values are specified. `PEXExtSetRendererAttributeMaskAll` and `PEXExtSetRendererAttributeMask` can be called to set up the extended value mask.

DATA STRUCTURES:

```
typedef XID        PEXRenderer;
```

See also `PEXlib.h` and `PEXExtlib.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.

BadPEXNameSet
The specified name set resource identified 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`, `PEXExtGetRendererAttributes`, `PEXExtSetRendererAttributeMask`, `PEXExtSetRendererAttributeMaskAll`, `PEXGetRendererDynamics`

t

Extended Set Lookup Table Entries

NAME:

PEXExtSetTableEntries - Extended Set Lookup Table Entries

SYNTAX:

void PEXExtSetTableEntries(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 the table entries to be set.
table_type	The type of lookup table entries to be set.
entries	An array of table entries.

RETURNS:

None

DESCRIPTION:

The function sets the lookup table entries in the specified extended 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 *table_type* is `PEXLUTTextFont`   <5.1 types>
...
...
`PEXPatternEntry` if *table_type* is `PEXLUTPattern`
`PEXExtTMBindingEntry` if *table_type* is `PEXExtLUTTMBinding`
<extended types>
`PEXExtTMCoordSourceEntry` if *table_type* is `PEXExtLUTTMCoordSource`
`PEXExtTMCompositionEntry` if *table_type* is `PEXExtLUTTMComposition`
`PEXExtTMSamplingEntry` if *table_type* is `PEXExtLUTTMSampling`

DATA STRUCTURES:

```
typedef XID       PEXLookupTable;
```

See also `PEXlib.h` and `PEXExtlib.h`

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.

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

SEE ALSO:

`PEXCreateRenderer`, `PEXExtGetRendererAttributes`, `PEXExtSetRendererAttributeMask`, `PEXExtSetRendererAttributeMaskAll`, `PEXGetRendererDynamics`

Extended Create Texture Map

NAME:

PEXExtCreateTM - Create a Texture Map

SYNTAX:

PEXExtTextureMap PEXExtCreateTM(Display **display*, int *domain*,
PEXExtTMDomainData **domain_data*, PEXExtTexelArray **texel_arrays*)

PARAMETERS:

display	A pointer to a display structure returned by a successful `XOpenDisplay` call.
domain	The texture map domain type.
domain_data	A pointer to domain information specifying the type of texture map to be created and the type of texel data.
texel_arrays	The texel data.

RETURNS:

The X resource identifier of the newly-created texture map.

DESCRIPTION:

This function converts the data described by the domain and texel array into an internal texture map resource. Once this function returns from its storage operation, all memory utilized by the texel arrays may be reclaimed for other use.

The *domain* and *domain_data* provide information global to the texture map to be created. The list of texel arrays provides the actual texel data. The number of arrays is determined by the number of levels and the texture map type. Texel arrays are ordered sequentially by logical levels, the base level being first in the list. Every subsequent map is ordered by its level from the largest dimension (t0, t1, or t2) down to the smallest dimension until all arrays have been defined. The texels are assumed to be stored in t0-t1-t2 order, that is t0 varies the fastest as texels are stored into the texel array.

DATA STRUCTURES:

```
typedef XID        PEXExtTextureMap;
```

See also `PEXlib.h` and `PEXExtlib.h`

ERRORS:

BadAlloc
The server failed to allocate the resource.

BadIDChoice
The specified ID already in use or not in range assigned to client.

BadMatch
The number of texel arrays given does not match the number of texel arrays expected.

BadPEXFloatingPointFormat
The specified floating point format is invalid or unsupported.

BadValue
A specified value is out of range.

SEE ALSO:

`PEXExtCreateFilteredTM`, `PEXExtCreateFilteredTMFromWindow`,
`PEXExtCreateTMDescription`

Extended Create Texture Map Description

NAME:

PEXExtCreateTMDescription - Create a Texture Map Description

SYNTAX:

PEXExtTMDescription PEXExtCreateTMDescription(Display **display*,
int *parameterization*, PEXExtTMParameterizationData **param_data*,
int *tm_rendering_order*, unsigned int *count*, PEXExtTextureMap **tm_ids*)

PARAMETERS:

display	A pointer to a display structure returned by a successful `XOpenDisplay` call.
parameterization	The texture map parameterization type.
param_data	A pointer to parameterization data.
tm_rendering_order	The texture mapping rendering order (`PEXExtTMRenderingOrderPreSpecular`, `PEXExtTMRenderingOrderPostSpecular`).
count	The number of texture map resource identifiers.
tm_ids	A pointer to a list of texture map resource identifiers.

RETURNS:

The X resource identifier of the newly-created texture map description.

DESCRIPTION:

This function converts the data described by the parameterization and texture map resources into an internal texture map description resource. Once this function returns from its storage operation, all memory may be reclaimed for other use.

DATA STRUCTURES:

```
typedef XID        PEXExtTMDescription;
```

See also `PEXlib.h` and `PEXExtlib.h`

ERRORS:

BadAlloc
The server failed to allocate the resource.

BadIDChoice
The specified ID already in use or not in range assigned to client.

BadPEXFloatingPointFormat
The specified floating point format is invalid or unsupported.

BadValue
A specified value is out of range.

SEE ALSO:

`PEXExtCreateTM`, `PEXExtCreateFilteredTM`,
`PEXExtCreateFilteredTMFromWindow`

t Extended Free Texture Map

NAME:

PEXExtFreeTM - Free a Texture Map Resource

SYNTAX:

void PEXExtFreeTM(Display **display*, PEXExtTextureMap *texture_map*)

PARAMETERS:

display	A pointer to a display structure returned by a successful `XOpenDisplay` call.
texture_map	The resource identifier of the texture map.

RETURNS:

None

DESCRIPTION:

This request deletes the association between the resource ID and the texture map. The texture map storage will be freed when no other resource references it.

DATA STRUCTURES:

```
typedef XID        PEXExtTextureMap;
```

See also `PEXlib.h` and `PEXExtlib.h`

ERRORS:

BadValue
The specified texture map ID is invalid or is an ID of a resource other than a texture map.

SEE ALSO:

`PEXExtCreateTM`

t Extended Free Texture Map Description

NAME:

PEXExtFreeTMDescription - Free a Texture Map Description Resource

SYNTAX:

void PEXExtFreeTMDescription(Display **display*, PEXExtTMDescription *tm_description*)

PARAMETERS:

display	A pointer to a display structure returned by a successful `XOpenDisplay` call.
tm_description	The resource identifier of the texture map description.

RETURNS:

None

DESCRIPTION:

This request deletes the association between the resource ID and the texture map description. The texture map storage will be freed when no other resource references it.

DATA STRUCTURES:

```
typedef XID        PEXExtTMDescription;
```

See also `PEXlib.h` and `PEXExtlib.h`

ERRORS:

BadValue
The specified *tm_description* is an invalid texture map description ID or is an ID of a resource other than a texture map description.

SEE ALSO:

`PEXExtCreateTMDescription`

Extended Get Pipeline Context Attributes

NAME:

PEXExtGetPipelineContext - Get Extended Pipeline Context Attributes

SYNTAX:

PEXExtPCAttributes *PEXExtGetPipelineContext(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 extended pipeline context.
value_mask	A pointer to an array of three unsigned long.

RETURNS:

A pointer to the extended pipeline context values; a null pointer if unsuccessful.

DESCRIPTION:

This function returns the requested attributes values of the extended pipeline context. The value mask indicates which attribute values are requested. `PEXSetPCAttributeMaskAll` can be called to set up the non-extended portion of the value mask and `PEXExtSetPCAttributeMask` can be called to set up any portion of the value mask. PEXlib allocates the memory for the return value. `PEXExtFreePCAttributes` should be called to deallocate the memory.

DATA STRUCTURES:

```
typedef XID        PEXPipelineContext;
```

See also `PEXlib.h` and `PEXExtlib.h`

ERRORS:

BadPEXFloatingPointFormat
The specified floating point format is invalid or unsupported.

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`, `PEXExtChangePipelineContext`, `PEXExtFreePCAttributes`

Extended Get Renderer Attributes

NAME:

PEXExtGetRendererAttributes - Get Extended Renderer Attribute Values

SYNTAX:

PEXExtRendererAttributes *PEXExtGetRendererAttributes(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 extended renderer.
value_mask	A pointer to an array of two unsigned long. Indicates which attributes are to be returned from the renderer.

RETURNS:

A pointer to the extended renderer attribute values; a null pointer if unsuccessful.

DESCRIPTION:

This function returns the requested attributes values of the extended renderer. The value mask indicates which attribute values are requested. `PEXExtSetRendererAttribute-MaskAll` and `PEXExtSetRendererAttributeMask` can be called to set up the extended value mask. PEXlib allocates the memory for the return value. `PEXExtFreeRendererAttributes` should be called to deallocate the memory.

DATA STRUCTURES:

```
typedef XID        PEXRenderer;
```

See also `PEXlib.h` and `PEXExtlib.h`

ERRORS:

BadPEXFloatingPointFormat
The specified floating point format is invalid or unsupported.

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`, `PEXExtChangeRenderer`, `PEXGetRendererDynamics`, `PEXExtFreeRendererAttributes`

t Extended Get Table Entry

NAME:

PEXExtGetTableEntry - Extended Get Table Entry

SYNTAX:

PEXPointer PEXExtGetTableEntry(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 (`PEXDefinedEntry` or `PEXDefaultEntry`).
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.

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` should be called to deallocated the memory.

DATA STRUCTURES:

```
typedef XID        PEXLookupTable;
```

See also `PEXlib.h` and `PEXExtlib.h`

ERRORS:

BadPEXFloatingPointFormat
The specified floating point format is invalid or unsupported.

BadPEXLookupTable
The specified lookup table resource identifier is invalid.

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

SEE ALSO:

`PEXExtSetTableEntries`, `PEXSetTableEntries`, `PEXFreeTableEntries` t **Extended Get Table Entries**

NAME:

PEXExtGetTableEntries - Extended Get Table Entries

SYNTAX:

Status PEXExtGetTableEntries(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.
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, starting at the specified entry index. 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.

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` should be called to deallocated the memory.

DATA STRUCTURES:

```
typedef XID        PEXLookupTable;
```

See also `PEXlib.h` and `PEXExtlib.h`

ERRORS:

BadPEXFloatingPointFormat
The specified floating point format is invalid or unsupported.

BadPEXLookupTable
The specified lookup table resource identifier is invalid.

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

SEE ALSO:

`PEXExtSetTableEntries`, `PEXSetTableEntries`, `PEXFreeTableEntries`

Extended Fetch Elements

NAME:

PEXExtFetchElements - Extended Fetch Elements

SYNTAX:

Status PEXExtFetchElements(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 of 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.

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.

A 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:

```
typedef XID        PEXStructure;
```

ERRORS:

BadPEXFloatingPointFormat
The specified floating point format is invalid or unsupported.

BadPEXStructure
The specified structure resource identifier is invalid.

BadValue
A specified value for whence parameter is invalid.

SEE ALSO:

`PEXCreateStructure`, `PEXExtDecodeOCs`, `PEXExtEncodeOCs`, `PEXSendOCs`

Extended Query Color Approximation

NAME:

PEXExtQueryColorApprox - Find a Supported Color Approximation for the Specified Example Drawable

SYNTAX:

int PEXExtQueryColorApprox(Display **display*, Drawable *drawable*,
PEXColorApproxEntry **color_approx*, unsigned long **count_return*,
PEXColorApproxEntry ***color_approx_return*, int **all_return*)

PARAMETERS:

display	A pointer to a display structure returned by a successful `XOpenDisplay` call.
drawable	The drawable indicates the depth, root, and visual (if the drawable has one) the application is interested in.
color_approx	A pointer to the desired color approximation. The application wants to determine if the desired color approximation is supported by windows of the same root and depth as the drawable.
count_return	Returns the number of returned alternative supported color approximations. Not valid if the function returns True.
color_approx_return	Returns a list of alternative supported color approximations. Not valid if the function returns True.
all_return	Returns True if the list of returned color approximations is the complete list of supported color approximations. Not valid if the function returns True.

RETURNS:

True if the given color approximation is supported; False otherwise. If an error condition was not identified and False is returned, the output parameters of the function provide alternative supported color approximation(s).

DESCRIPTION:

This function is used to find a supported color approximation for the given example drawable. If the given color approximation is supported by drawables of the same visual (if the drawable has one), root, and depth as the given example drawable, the function returns True.

If the given color approximation is not supported, the function returns False. In this case, a list of color approximations which are supported by drawables of the same visual (if the drawable has one), root, and depth as the given example drawable are returned. It is indicated whether this list is exhaustive.

PEXlib allocates memory if any alternative supported color approximations are supported. If PEXlib is unable to allocate the memory it requires, parameter *count_return* will be returned set to zero and parameter *color_approx_return* will be returned set to NULL. `XFree` should be called to deallocate the memory if it was allocated.

DATA STRUCTURES:

See the existing PEXlib 5.1 data structure definitions.

ERRORS:

BadPEXFloatingPointFormat
The specified floating point format is invalid or unsupported.

BadPEXDrawable
The specified drawable resource identifier is invalid.

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

BadValue
A specified value is out of range.

SEE ALSO:

`PEXSetTableEntries`, `PEXExtSetTableEntries`

t Extended Set Texture Mapping Perspective Correction

NAME:

PEXExtSetTMPerspectiveCorrection - Set Texture Mapping Perspective Correction

SYNTAX:

void PEXExtSetTMPerspectiveCorrection(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	Type of texture mapping perspective correction to apply (`PEXExtTMPerspCorrectNone`, `PEXExtTMPerspCorrectVertex`, `PEXExtTMPerspCorrectPixel`).

RETURNS:

None

DESCRIPTION:

When processed by a renderer, this command will modify the renderer's *tm_perspective_correction* attribute. If the specified type is not supported by the PEX server, texture mapping perspective correction `PEXExtTMPerspCorrectNone` is used. The default value is `PEXExtTMPerspCorrectNone`.

Supported values for texture map perspective correction are inquirable using `PEXGetEnumTypeInfo`.The default texture map perspective correction is `PEXExtTMPerspCorrectNone`.

ERRORS:

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO:

`PEXExtSetTMSampleFrequency`

t Extended Set Texture Mapping Sample Frequency

NAME:

PEXExtSetTMSampleFrequency - Set Texture Mapping Sample Frequency

SYNTAX:

void PEXExtSetTMSampleFrequency(Display **display*, XID *resource_id*,
PEXOCRequestType *req_type*, int *frequency*)

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`).
frequency	The texture mapping sample frequency (`PEXExtTMSampleFrequencyPixel`, `PEXExtTMSampleFrequencyInterpDep`).

RETURNS:

None

DESCRIPTION:

When processed by a renderer, this command will modify the renderer's *tm_sample_frequency* attribute. If the specified type is not supported by the PEX server, texture mapping sample frequency `PEXExtTMSampleFrequencyPixel` is used.

Supported values for texture map sample frequency are inquirable using `PEXGetEnumTypeInfo`.The default texture map sample frequency is `PEXExtTMSampleFrequencyPixel`.

ERRORS:

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO:

`PEXExtSetTMPerspectiveCorrection`

t Extended Set Texture Mapping Resource Hints

NAME:

PEXExtSetTMResourceHints - Set Texture Mapping Resource Hints

SYNTAX:

void PEXExtSetTMResourceHints(Display **display*, XID *resource_id*,
PEXOCRequestType *req_type*, int *optimization_hint*, unsigned int *count*,
PEXTableIndex **priorities*)

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`).
optimization_hint	Resource optimization approach to consider for all subsequently activated textures (`PEXExtTMResourceHintNone`, `PEXExtTMResourceHintSpeed`, `PEXExtTMResourceHintSpace`).
count	The number of indices.
priorities	An array of texture binding table indices.

RETURNS:

None

DESCRIPTION:

When processed by a renderer, this command will modify the renderer's *tm_resource_hints* attributes. The list of prioritized textures replaces the previous list. If a referenced TM Binding LUT entry is not defined, it is not used in resource optimization by the server.

In the pipeline context, the default *tm_resource_hints* optimization is None and the default *tm_resource_hints* prioritized textures is the empty list.

ERRORS:

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

Extended Set Current Active Texture Maps

NAME:

PEXExtSetActiveTextures - Set Current Active Texture Maps

SYNTAX:

void PEXExtSetActiveTextures(Display **display*, XID *resource_id*,
PEXOCRequestType *req_type*, unsigned int *count*,
PEXTableIndex **textures*)

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	Number of textures to activate.
textures	An array of texture binding table indices corresponding to the texture maps to activate.

RETURNS:

None

DESCRIPTION:

When processed by a renderer, this command will modify the renderer's *active_textures* attribute. This list of textures replaces the previous list. If a referenced TM Binding LUT entry is not defined, the default entry will be used instead. If the default entry is not defined, the default values for the TM Binding LUT will be applied (this results to a black and white checkerboard texture being applied). The default pipeline context value is the empty list.

The textures in the active texture list are sequentially applied in the order provided using the appropriate texture map parameterization, sampling, and composition methods specified in the TM Binding LUT. If the number of textures to be activated exceeds `PEXExtIDMaxTextureMaps`, the list of textures in beyond the first '`PEXExtIDMaxTextureMaps`' textures will be ignored.

Setting active textures with a count of zero means there are no active textures, resulting in the default texture being applied.

A texture is activated 1) if it is not already activated, and 2) when the `PEXExtOCActive-Textures` or `PEXExtOCBFActiveTextures` OC is processed with the texture in the activation list. A texture will be deactivated when it is no longer in the *active_textures* or *bf_active_textures* list.

ERRORS:

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO:

`PEXSetBFActiveTextures`

Extended Set Current Back-Face Active Texture Maps

NAME:

PEXExtSetBFActiveTextures - Set Current Back-Face Active Texture Maps

SYNTAX:

void PEXExtSetBFActiveTextures(Display **display*, XID *resource_id*,
PEXOCRequestType *req_type*, unsigned int *count*,
PEXTableIndex **textures*)

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	Number of textures to activate.
textures	An array of texture binding table indices corresponding to the texture maps to activate.

RETURNS:

None

DESCRIPTION:

When processed by a renderer, this command will modify the renderer's *bf_active_textures* attribute. This list of textures replaces the previous list. If a referenced TM Binding LUT entry is not defined, the default entry will be used instead. If the default entry is not defined, the default values for the TM Binding LUT will be applied (this results to a black and white checkerboard texture being applied). The default pipeline context value is the empty list.

The textures in the active texture list are sequentially applied in the order provided using the appropriate texture map parameterization, sampling, and composition methods specified in the TM Binding LUT. If the number of textures to be activated exceeds `PEXExtIDMaxTextureMaps`, the list of textures in beyond the first '`PEXExtIDMaxTextureMaps`' textures will be ignored.

Setting active textures with a count of zero means there are no active textures, resulting in the default texture being applied.

A texture is activated 1) if it is not already activated, and 2) when the `PEXExtOCActive-Textures` or `PEXExtOCBFActiveTextures` OC is processed with the texture in the activation list. A texture will be deactivated when it is no longer in the *active_textures* or *bf_active_textures* list.

ERRORS:

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO:

`PEXSetActiveTextures`

t Extended Fill Area Set With Data

NAME:

PEXExtFillAreaSetWithData - 3D Set of Fill Areas Primitive With Additional Data

SYNTAX:

void PEXExtFillAreaSetWithData(Display **display*, XID *resource_id*,
PEXOCRequestType *req_type*, int *shape_hint*, int *ignore_edges*,
int *contour_hint*, unsigned int *vertex_fp_data_size*,
unsigned int *facet_attributes*, unsigned int *vertex_attributes*,
int *color_type*, unsigned int *count*, PEXFacetData **facet_data*,
PEXExtListOfVertex **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`).
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`).
vertex_fp_data_size	The number of floating point data values defined with each vertex.
facet_attributes	A mask indicating the facet attributes provided (`PEXGANone`, `PEXGAColor`, `PEXGANormal`).
vertex_attributes	A mask indicating the vertex attributes provided (`PEXGANone`, `PEXGAColor`, `PEXGANormal`, `PEXGAEdges`, `PEXExtGAData`).

color_type	The type of color data provided (`PEXColorTypeIndexed`, `PEXColorTypeRGB`, `PEXColorTypeCIE`, `PEXColorTypeHSV`, `PEXColorTypeHLS`, `PEXColorTypeRGB8`, `PEXColorTypeRGB16`).
count	The number of fill areas in the set.
facet_data	A pointer to the facet data.
vertex_lists	A pointer to the list of vertex arrays defining each contour of the fill area set.

RETURNS:

None

DESCRIPTION:

When processed by a renderer, this command will cause a fill area set to be rendered. The behavior of this primitive is identical to that of the 3D fill area set primitive, except additional information can be specified for each of the fill area sets, for each edge, and for each vertex. Color values that are passed will be of the type specified by *color_type*.

The parameter *facet_attributes* indicates the attributes which are specified in the *facet_data* parameter. The components of the facet attributes bitmask are, in order:

color	COLOR
normal	VECTOR_3D

If any of the attribute bits is set, the corresponding attributes must be present in the *facet_data* parameter and must be passed in the order in which they appear above.

The parameter *vert_attributes* specifies the attributes which are specified at each fill area set vertex. The components of the vertex attributes mask are, in order:

color	COLOR
normal	VECTOR_3D
edges	SWITCH
fp_data	LISTofFLOAT

If any of the attribute bits is set, the corresponding attributes must be present for each vertex, and they must be passed after the coordinate data for each vertex in the order in which they appear in the list above.

If color values are passed as part of *per_facet_data* or as part of the *per_vert_data* list, they are considered to be part of the primitive and are used instead of the *surface_color* attribute.

Vertex colors will be utilized rather than facet colors if both are provided. In addition, the use of per-vertex colors is affected by the *surface_interp* attribute, which is obtained directly from the *surface_interp* value if the *surface_interp_asf* attribute is set to Individual or from the *interior_bundle_index*'th entry in the renderer's *interior_bundle*. The *surface_interp* attribute defines how color values between the vertices are to be computed.

If normals are passed per vertex, they are taken to be the normals at the vertices of the fill area. All normals are assumed to be unit vectors. The effect of rendering with vertex normals which are not unit vectors is implementation-dependent.

If surface edge flags are specified per vertex, each flag specifies whether to draw the edge from the vertex with which the flag is specified to the next vertex. (E.g., for a facet with four vertices, the edge flag associated with vertex #1 indicates whether to draw edge #1-#2, edge flag #2 specifies edge #2-#3, edge flag #3 specifies edge #3-#4, and edge flag #4 specifies edge #4-#1.) Surface edges are always drawn with the surface edge color, never with per facet or per vertex colors.

At structure creation time, if any of the contours of the fill area set has fewer than three vertices, or if there are no contours defined, the output command is stored in the structure, but when this output command is interpreted, it has no visual effect. In immediate mode, such a primitive is ignored.

An `OutputCommand` error is reported if the shape, *ignore_edges*, or *contour_hint* parameter is invalid.

If *vertex_fp_data_size* is non-zero, you will need to pack the vertex data yourself. The coordinate data is packed first, followed by the color, normal, and edge data (if present), in exactly the same order as defined by data structure definitions `PEXCoord` through `PEXVertexRGB16NormalEdge`. The floating point values directly follow.

For example, if an application has the coordinates, normal, and three floating point data list values for a vertex, it would pack the data as follows:

```
typedef struct {
    /*  Use an existing PEXArrayOfVertex member and add the floating
      *  point data list:  */
    PEXVertexNormalcoords_and_normal;
    floatfp_data[3];
} MyPEXVertexNormalFPData3;

MyPEXVertexNormalFPData3 my_vertex_data;
PEXExtArrayOfVertex vertex_data;

my_vertex_data.coords_and_normal.point.x = MY_VERTEX_X;
my_vertex_data.coords_and_normal.point.y = MY_VERTEX_Y;
my_vertex_data.coords_and_normal.point.z = MY_VERTEX_Z;
my_vertex_data.coords_and_normal.normal.x = MY_VERTEX_NORMAL_X;
```

```
my_vertex_data.coords_and_normal.normal.y = MY_VERTEX_NORMAL_Y;
my_vertex_data.coords_and_normal.normal.z = MY_VERTEX_NORMAL_Z;
my_vertex_data.fp_data[0] = MY_VERTEX_FP_DATA_VALUE_1;
my_vertex_data.fp_data[1] = MY_VERTEX_FP_DATA_VALUE_2;
my_vertex_data.fp_data[2] = MY_VERTEX_FP_DATA_VALUE_3;

vertex_data.with_fp_data = (PEXPointer)(&my_vertex_data);
```

DATA STRUCTURES:

See also `PEXlib.h` and `PEXExtlib.h`

ERRORS:

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO:

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

t Extended Set of Fill Area Sets

NAME:

PEXExtSetOfFillAreaSets - 3D Set of Fill Area Sets Primitive

SYNTAX:

void PEXExtSetOfFillAreaSets(Display **display*, XID *resource_id*,
PEXOCRequestType *req_type*, int *shape_hint*,
unsigned int *vertex_fp_data_size*,
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*,
PEXExtArrayOfVertex *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`).
vertex_fp_data_size	The number of floating point data values defined per vertex fill area set attributes.
facet_attributes	A mask indicating the facet attributes provided (`PEXGANone`, `PEXGAColor`, `PEXGANormal`).
vertex_attributes	A mask indicating the vertex attributes provided (`PEXGANone`, `PEXGAColor`, `PEXGANormal`, `PEXExtGAData`).
edge_attributes	A mask indicating the edge attributes provided (`PEXGANone`, or `PEXGAEdges`).

contour_hint	A flag that indicates 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.

RETURNS:

None

DESCRIPTION:

When processed by a renderer, this command will draw a set of fill area sets that may be connected (i.e., individual fill area sets may share geometry and attribute information at vertices). Shading calculations and transformations need only be performed once per shared vertex instead of once for every fill area set that shares the vertex. Similarly, data can be transmitted across the network once per unique vertex instead of once for every fill area set sharing the vertex.

The shape parameter is passed as a hint as to the type of contours that comprise each of the fill area sets. A shape hint of Unknown means nothing is known about the shape of the constituent contours. A shape of Complex means some contours of the fill area sets may have edges that self-intersect. A shape of Nonconvex means none of the contours of the fill area sets have edges that self-intersect, but some may not be wholly convex. Convex means all of the interior angles of all of the contours are convex. (Note a fill area set with more than one contour is always allowed to have contours that intersect. It is quite possible the only times the fastest rendering code path can be taken are if the number of contours in a fill area set is equal to one or if *contours_all_1* is True, and the shape flag for the set of fill area sets primitive is Convex.) Contours of a higher complexity than indicated by their shape hint are drawn in an implementation-dependent manner. PEX server extensions may ignore the shape hint and treat all constituent contours as shape Unknown.

The *contour_hint* parameter provides further information as to the relationship between contours in each of the fill area sets. If *contour_hint* is Disjoint, all contours will be spatially disjoint. No overlapping or intersection occurs between any contours in any of the fill area sets. If *contour_hint* is Nested, 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 *contour_hint* is Intersecting, separate contours may have edges which are coincident or overlap. If *contour_hint* is Unknown, nothing is known about the interrelationships between contours. Fill area sets with contours that have higher complexity interrelationships than indicated by their contour hint are drawn in an implementation-dependent manner. PEX server extensions may ignore the contour hint and treat all fill area sets as Unknown.

Color values which are passed in will be of the type specified by *color_type*.

The parameter *fas_attributes* indicates the attributes which are specified for each fill area set. The components of the *fas_attributes* bitmask are, in order:

color	COLOR
normal	VECTOR_3D

If any of the attribute bits is set, the corresponding attribute must be present in *per_fas_data*, which contains one data record for each fill area set. The attributes passed in this way must be passed in the order in which they appear in the list above. If *fas_attributes* is null, the *per_fas_data* list will be null as well. Otherwise there will be *num_fas* entries in *per_facet_data*.

The parameter *vert_attributes* indicates the attributes which are provided with each vertex in the list of unique vertices specified by *per_vert_data*. The components of the vertex attributes bitmask are, in order:

color	COLOR
normal	VECTOR_3D
fp_data	LISTofFLOAT

If any of the attribute bits is set, the corresponding attribute must be present for each vertex in the *per_vert_data* list, and it must be passed after the coordinate data for each vertex. The attributes which are passed in this way must be passed in the order in which they appear in the list above. There will always be *num_verts* entries in the *per_vert_data* list.

If color values are passed as part of *per_facet_data* or as part of the *per_vert_data* list, they are considered to be part of the primitive and are used instead of the *surface_color* attribute. Vertex colors will be utilized rather than facet colors if both are provided. In addition, the use of per-vertex colors is affected by the *surface_interp* attribute, which is obtained directly from the *surface_interp* value if the *surface_interp_asf* attribute is set to Individual or from the *interior_bundle_index*'th entry in the renderer's *interior_bundle*. The *surface_interp* attribute defines how color values between the vertices are to be computed.

All normals are assumed to be unit vectors. The effect of rendering with facet or vertex normals which are not unit vectors is implementation dependent.

The parameter *edge_attributes* specifies the attributes that are specified at each edge. The components of the edge attributes bitmask are, in order:

edges	SWITCH

If any of the attribute bits is set, the corresponding attribute must be present for each edge in the list *per_edge_data*. If none of the attribute bits are set, the list *per_edge_data* will be an empty list. Otherwise, it will contain *num_edges* entries, each of which contains a switch indicating whether the corresponding edge should be drawn. This list is a flattened list without counts, so if it is nonempty, it is up to the PEX server extension to maintain a pointer to the proper position in this list while processing the data in contours. If edge switches are supplied, each flag specifies whether to draw the edge from the vertex with which the flag is specified to the next vertex. (e.g., for a contour with four vertices, the edge flag associated with vertex #1 indicates whether to draw edge #1-#2, edge flag #2 specifies edge #2-#3, edge flag #3 specifies edge #3-#4, and edge flag #4 specifies edge #4-#1.) Surface edges are always drawn with the surface edge color, never with per-fill-area-set or per-vertex colors.

The connectivity of the primitive is defined by the contours array. The number of contours in the first fill area set is contained as the first entry in the contours array. The number of contours is followed by a list of data recoreds, one for each of the contours in the fill rea set. If this number is n, then the next n data records in the contours array are used to define the first fill area set. The value following the contour count contains the number of vertices in the first contour of the first fill area set. If this number is m, then the next m values in the array comprise the data record for the first contour. This data record contains the indices for the vertices of the fist contour. Depending on n, the next value in the list is either the number of vertices in the second contour for the first fill ara set, or it is the number of contours in the second fill area set. As a special case, if *contours_all_1* is True, then the contour count field in each fill area set is guaranteed to be one.

Vertices are numbered with indices starting from zero (i.e., the first vertex is referenced as vertex 0).

All attributes which affect the representation of fill area sets also affect the representation of the set of fill area sets primitive.

An OutputCommand error is reported if any of the edge flags in the *per_edge_data* array is not On or Off, or if any of the shape, *contour_hint*, or *contours_all_1* parameters is invalid, or if any of the vertex indices in the contours array is out of range, or if there are not as many edge flags as there are vertex indices in the lists in the contours array.

DATA STRUCTURES:

See also `PEXlib.h` and `PEXExtlib.h`

ERRORS:

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO:

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

t

Extended Triangle Strip

NAME:

PEXExtTriangleStrip - 3D Triangle Strip Primitive

SYNTAX:

void PEXExtTriangleStrip(Display **display*, XID *resource_id*,
PEXOCRequestType *req_type*, unsigned int *vertex_fp_data_size*,
unsigned int *facet_attributes*, unsigned int *vertex_attributes*,
int *color_type*, PEXArrayOfFacetData *facet_data*,
unsigned int *count*, PEXExtArrayOfVertex *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`).
vertex_fp_data_size	The number of floating point data values defined per vertex.
facet_attributes	A mask indicating the facet attributes provided (`PEXGANone`, `PEXGAColor`, `PEXGANormal`).
vertex_attributes	A mask indicating the vertex attributes provided (`PEXGANone`, `PEXGAColor`, `PEXGANormal`, `PEXExtGAData`).
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:

When processed by a renderer, this command will cause a triangle strip primitive to be drawn. Color values that are passed will be of the type specified by *color_type*. The parameter *facet_attributes* indicates the attributes which are specified for each facet of the triangle strip. The components of the facet attributes bitmask are, in order:

color	COLOR
normal	VECTOR_3D

If any of the attribute bits is set, the corresponding attribute must be present in *facet_data*, which is the data that defines each triangular facet. The attributes that are passed in this way must be passed in the order in which they appear in the list above. There will be n-2 entries in the *facet_data* list, where n is the number of entries in the vertices list.

The parameter *vert_attributes* specifies the attributes which are specified at each triangle strip vertex. The components of the vertex attributes bitmask are, in order:

color	COLOR
normal	VECTOR_3D
fp_data	LISTofFLOAT

If any of the attribute bits is set, the corresponding attribute must be present for each vertex, and it must be passed after the coordinate data for each vertex. The attributes which are passed in this way must be passed in the order in which they appear in the list above.

If color values are passed as part of *per_facet_data* or as part of the *per_vert_data* list, they are considered to be part of the primitive and are used instead of the *surface_color* attribute. Vertex colors will be utilized rather than facet colors if both are provided. In addition, the use of per-vertex colors is affected by the *surface_interp* attribute, which is obtained directly from the *surface_interp* value if the *surface_interp_asf* attribute is set to Individual or from the *interior_bundle_index*'th entry in the renderer's *interior_bundle*. The *surface_interp* attribute defines how color values between the vertices are to be computed.

All normals are assumed to be unit vectors. The effect of rendering with vertex normals which are not unit vectors is implementation-dependent.

The triangle strip is created from the vertex array. The strip is composed of n-2 triangles, where n is the number of vertices. The first triangle 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.

All attributes which affect the representation of fill area sets also affect the representation of the triangle strip primitive.

At structure creation time, if the triangle strip has fewer than three vertices, the output command is stored in the structure, but when this output command is interpreted, it has no visual effect. In immediate mode, such a primitive is ignored.

DATA STRUCTURES:

See also `PEXlib.h` and `PEXExtlib.h`

ERRORS:

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO:

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

t

Extended Quadrilateral Mesh

NAME:

PEXExtQuadrilateralMesh - 3D Quadrilateral Mesh Primitive

SYNTAX:

void PEXExtQuadrilateralMesh(Display **display*, XID *resource_id*,
PEXOCRequestType *req_type*, int *shape_hint*,
unsigned int *vertex_fp_data_size*,
unsigned int *facet_attributes*, unsigned int *vertex_attributes*,
int *color_type*, PEXArrayOfFacetData *facet_data*,
unsigned int *col_count*, unsigned int *row_count*,
PEXExtArrayOfVertex *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 which describes all of the contours (`PEXShapeComplex`, `PEXShapeNonConvex`, `PEXShapeConvex`, `PEXShapeUnknown`).
vertex_fp_data_size	The number of floating point data values defined per vertex.
facet_attributes	A mask indicating the facet attributes provided (`PEXGANone`, `PEXGAColor`, `PEXGANormal`).
vertex_attributes	A mask indicating the vertex attributes provided (`PEXGANone`, `PEXGAColor`, `PEXGANormal`, `PEXExtGAData`).
color_type	The type of color data provided (`PEXColorTypeIndexed`, `PEXColorTypeRGB`, `PEXColorTypeCIE`, `PEXColorTypeHSV`, `PEXColorTypeHLS`, `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 quadrilateral mesh.

RETURNS:

None

DESCRIPTION:

When processed by a renderer, this command will cause a quadrilateral mesh primitive to be rendered. The shape parameter is passed as a hint as to the type of quadrilaterals which comprise the primitive. A shape hint of Unknown means nothing is known about the shape of the constituent quadrilaterals. A shape of Complex means some of the quadrilaterals may have edges which self-intersect. A shape of Nonconvex means none of the quadrilaterals have edges which self-intersect, but some may not be wholly convex. Convex means all of the interior angles of all of the quadrilaterals are convex. Quadrilaterals of a higher complexity than indicated by their shape hint are drawn in an implementation-dependent manner. Color values which are passed will be of the type specified by *color_type*.

The parameter *facet_attributes* indicates the attributes which are specified for each facet of the quadrilateral mesh. The components of the facet attributes bitmask are, in order:

color	COLOR
normal	VECTOR_3D

If any of the attribute bits is set, the corresponding attribute must be present in *facet_data*, which is the data that defines each quadrilateral facet. The attributes which are passed in this way must be passed in the order in which they appear in the list above.

The parameter *vert_attributes* specifies the attributes which are specified at each quadrilateral mesh vertex. The components of the vertex attributes bitmask are, in order:

color	COLOR
normal	VECTOR_3D
fp_data	LISTofFLOAT

If any of the attribute bits is set, the corresponding attribute must be present for each vertex, and it must be passed after the coordinate data for each vertex. The attributes which are passed in this way must be passed in the order in which they appear in the list above.

If color values are passed as part of *per_facet_data* or as part of the *per_vert_data* list, they are considered to be part of the primitive and are used instead of the *surface_color* attribute. Vertex colors will be utilized rather than facet colors if both are provided. In addition, the use of per-vertex colors is affected by the *surface_interp* attribute, which is obtained directly from the *surface_interp* value if the *surface_interp_asf* attribute is set to Individual or from the *interior_bundle_index*'th entry in the renderer's *interior_bundle*. The *surface_interp* attribute defines how color values between the vertices are to be computed.

All normals are assumed to be unit vectors. The effect of rendering with vertex normals which are not unit vectors is implementation-dependent.

The surface will be created from a vertex array which is stored in row major order (i.e., the column number varies fastest as vertices are stored in the array). The (ith,jth), (i+1th,jth), (i+1th,j+1th) and (ith,j+1th) vertices are connected to create a single facet. Adjacent vertices are interconnected until the entire facet network is processed. There are *m_pts* x *n_pts* entries in the vertices array, and there are (m_pts-1) x (n_pts-1) entries in the *facet_data* array if any per-facet attributes are passed. *m_pts* is the number of columns in the vertex array and *n_pts* is the number of rows.

It is allowable for the boundary of a single facet to not reside in a single plane. The treatment of the vertex attributes in this case is implementation-dependent.

All attributes which affect the representation of fill area sets also affect the representation of the quadrilateral mesh primitive.

At structure creation time, if either *m_pts* or *n_pts* is less than two, the output command is stored in the structure, but when this output command is interpreted, it has no visual effect. In immediate mode, such a primitive is ignored.

An `OutputCommand` error is reported if the shape parameter is invalid.

DATA STRUCTURES:

See also `PEXlib.h` and `PEXExtlib.h`

ERRORS:

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO:

`PEXSetInteriorStyle`, `PEXSetInteriorStyleIndex`, `PEXSetSurfaceColorIndex`, `PEXSetSurfaceColor`, `PEXSetReflectionAttributes`, `PEXSetReflectionModel`, `PEXSetSurfaceInterpMethod`, `PEXSetBFInteriorStyle`, `PEXSetBFInteriorStyleIndex`, `PEXSetBFSurfaceColorIndex`,

PEXSetBFSurfaceColor, PEXSetBFReflectionAttributes,
PEXSetBFReflectionModel, PEXSetBFSurfaceInterpMethod,
PEXSetFacetCullingMode, PEXSetFacetDistinguishFlag,
PEXSetPatternSize, PEXSetPatternAttributes,
PEXSetPatternAttributes2D, PEXSetInteriorBundleIndex,
PEXSetSurfaceEdgeFlag, PEXSetSurfaceEdgeType,
PEXSetSurfaceEdgeWidth, PEXSetSurfaceEdgeColor,
PEXSetSurfaceEdgeColorIndex, PEXSetEdgeBundleIndex

Extended Set Primitive Anti-Aliasing

NAME:

PEXExtSetPrimitiveAA - Set Primitive Anti-Aliasing

SYNTAX:

void PEXExtSetPrimitiveAA(Display **display*, XID *resource_id*,
    PEXOCRequestType *req_type*, int *mode*, int *blend_op*)

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 primitive anti-aliasing mode (`PEXExtPrimAANone`, `PEXExtPrimAAPoint`, `PEXExtPrimAAVector`, `PEXExtPrimAAPointVector`, `PEXExtPrimAAPolygon`, `PEXExtPrimAAPointPolygon`, `PEXExtPrimAAVectorPolygon`, or `PEXExtPrimAAPointVectorPolygon`).
blend_op	The primitive anti-aliasing blend operation (`PEXExtPrimAABlendOpImpDep` or `PEXExtPrimAABlendOpSimpleAlpha`).

RETURNS:

None

DESCRIPTION:

When processed by a renderer, this command will modify the renderer's *primitive_aa* mode and *primitive_aa blend_op* attributes. If the specified mode is not supported by the server, primitive anti-aliasing mode `PEXExtPrimAANone` is used. If the specified blend operation is not supported by the server, primitive anti-aliasing blend `PEXExtPrimAABlendOpImpDep` is used. In the default pipeline context, the primitive anti-aliasing mode is `PEXExtPrimAANone` and the primitive anti-aliasing blend operation is `PEXExtPrimAABlendOpImpDep`.

Any integer may be specified as the primitive anti-aliasing mode or the primitive anti-aliasing blend operation in this output command. If an inconsistent value is specified in the data record, an `OutputCommand` error is reported.

Primitive anti-aliasing interacts with `HLHSR` in an implementation-dependent manner.

ERRORS:

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO:

`PEXGetEnumTypeInfo`

Extended Set Line Cap Style

NAME:

PEXExtSetLineCapStyle - Set Line Cap Style

SYNTAX:

void PEXExtSetLineCapStyle(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 cap style to be applied to wide lines (`PEXExtLineCapStyleButt`, `PEXExtLineCapStyleRound`, or `PEXExtLineCapStyleProjecting`).

RETURNS:

None

DESCRIPTION:

When processed by a renderer, this command will modify the renderer's *line_cap_style* attribute. If the specified line cap style is not supported by the server, line cap style `PEXExtLineCapStyleButt` is used. In the default pipeline context, the line cap style is `PEXExtLineCapStyleButt`.

Any integer value may be specified as the line cap style in this output command.

ERRORS:

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO:

`PEXGetEnumTypeInfo`

Extended Set Line Join Style

NAME:

PEXExtSetLineJoinStyle - Set Line Join Style

SYNTAX:

void PEXExtSetLineJoinStyle(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 join style to be applied to wide lines (PEXExtLineJoinStyleImpDep, PEXExtLineJoinStyleRound, PEXExtLineJoinStyleMiter, or PEXExtLineJoinStyleBevel).

RETURNS:

None

DESCRIPTION:

When processed by a renderer, this command will modify the renderer's *line_join_style* attribute. If the specified line join style is not supported by the server, line join style `PEXExtLineJoinStyleImpDep` is used. In the default pipeline context, the line cap style is `PEXExtLineJoinStyleImpDep`.

Any integer value may be specified as the line join style in this output command.

ERRORS:

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO:

`PEXGetEnumTypeInfo`

Extended 3D Ellipse

NAME:

PEXExtEllipse - 3D Ellipse

SYNTAX:

void PEXExtEllipse(Display **display*, XID *resource_id*, PEXOCRequest *req_type*, PEXCoord **center*, PEXVector **major*, PEXVector **minor*)

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`).
center	The center of the ellipse in modelling coordinates.
major	The major axis of the ellipse in modelling coordinates.
minor	The minor axis of the ellipse in modelling coordinates.

RETURNS:

None

DESCRIPTION:

When processed by a renderer, this command will cause an ellipse to be rendered. The ellipse's center and axes are defined in three dimensional modelling coordinate space. The major axis defines the positive X-axis of the ellipse local coordinate system. The minor axis is used in defining the positive Y-axis of the ellipse local coordinate system.

The major and minor axis parameters define a rectangle which circumscribes the ellipse and is anchored at the center of the ellipse. Two sides are parallel to the major axis. The other two sides are parallel to the minor axis. The major and minor axes extend from the center of the ellipse to midpoints of sides of the rectangle. The ellipse is tangent to the sides of the rectangle at their midpoints. The behavior if the major and minor axes are not orthogonal is device-dependent.

The current values of the fill area attributes are applied to the primitive. The ellipse is rendered according to the curve approximation criteria.

ERRORS:

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO:

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

Extended 2D Ellipse

NAME:

PEXExtEllipse2D - 2D Ellipse

SYNTAX:

void PEXExtEllipse2D(Display **display*, XID *resource_id*, PEXOCRequest *req_type*,
PEXCoord2D **center*, PEXVector2D **major*, PEXVector2D **minor*)

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`).
center	The center of the ellipse in 2D modelling coordinates.
major	The major axis of the ellipse in 2D modelling coordinates.
minor	The minor axis of the ellipse in 2D modelling coordinates.

RETURNS:

None

DESCRIPTION:

When processed by a renderer, this command will cause an ellipse to be rendered. The ellipse's center and axes are defined in two dimensional modelling coordinate space. The major axis defines the positive X-axis of the ellipse local coordinate system. The minor axis is used in defining the positive Y-axis of the ellipse local coordinate system. The ellipse, centered about the center coordinates, will lie in the modelling coordinate XY plane (Z = 0).

The major and minor axis parameters define a rectangle which circumscribes the ellipse and is anchored at the center of the ellipse. Two sides are parallel to the major axis. The other two sides are parallel to the minor axis. The major and minor axes extend from the center of the ellipse to midpoints of sides of the rectangle. The ellipse is tangent to the sides of the rectangle at their midpoints. The behavior if the major and minor axes are not orthogonal is device-dependent.

The current values of the fill area attributes are applied to the primitive. The ellipse is rendered according to the curve approximation criteria.

ERRORS:

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO:

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

t Extended 2D Circle

NAME:

PEXExtCircle2D - 2D Circle

SYNTAX:

void PEXExtCircle2D(Display **display*, XID *resource_id*,
PEXOCRequest *req_type*, PEXCoord2D **center*, double *radius*)

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`).
center	The center of the circle in 2D modelling coordinates.
radius	The radius of the circle in modelling coordinates.

RETURNS:

None

DESCRIPTION:

When processed by a renderer, this command will cause a circle to be rendered. The circle's center and radius are defined in two dimensional modelling coordinate space. A circle having the specified radius is drawn centered about the center coordinates in the modelling coordinate Z=0 plane.

The current values of the fill area attributes are applied to the primitive. The circle is rendered according to the curve approximation criteria.

If the radius of the circle is less than or equal to zero, a dot at the center will be displayed.

ERRORS:

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO:

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

Extended 3D Elliptical Arc

NAME:

PEXExtEllipticalArc - 3D Elliptical Arc

SYNTAX:

void PEXExtEllipticalArc(Display **display*, XID *resource_id*,
PEXOCRequest *req_type*, PEXCoord **center*, PEXVector **major*,
PEXVector **minor*, double *start*, double *stop*)

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`).
center	The center of the elliptical arc in modelling coordinates.
major	The major axis of the elliptical arc in modelling coordinates.
minor	The minor axis of the elliptical arc in modelling coordinates.
start	The start angle in radians.
stop	The stop angle in radians.

RETURNS:

None

DESCRIPTION:

When processed by a renderer, this command will cause an elliptical arc to be rendered. The elliptical arc's center and axes are defined in three dimensional modelling coordinate space. The major axis defines the positive X-axis of the ellipse local coordinate system. The minor axis is used in defining the positive Y-axis of the ellipse local coordinate system. A start and end angle relative to the positive X axis define the portion of the circumference drawn to form the specified arc.

The arc is drawn in a counter clockwise direction from the starting angle position to the ending angle position.

The major and minor axis parameters define a rectangle which circumscribes the ellipse and is anchored at the center of the ellipse. Two sides are parallel to the major axis. The other two sides are parallel to the minor axis. The major and minor axes extend from the center of the ellipse to midpoints of sides of the parallelogram. The ellipse is tangent to the sides of the parallelogram at their midpoints. The current values of the line attributes are applied to the primitive. The elliptical arc is rendered according to the curve approximation criteria.

ERRORS:

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO:

`PEXSetLineType`, `PEXSetLineWidth`, `PEXSetLineColorIndex`, `PEXSetLineColor`, `PEXSetPolylineInterpMethod`, `PEXSetLineBundleIndex`, `PEXSetCurveApprox`

t

Extended 2D Elliptical Arc

NAME:

PEXExtEllipticalArc2D - 2D Elliptical Arc

SYNTAX:

void PEXExtEllipticalArc2D(Display **display*, XID *resource_id*,
PEXOCRequest *req_type*, PEXCoord2D **center*, PEXVector2D **major*,
PEXVector2D **minor*, double *start*, double *stop*)

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`).
center	The center of the elliptical arc in2D modelling coordinates.
major	The major axis of the elliptical in 2D modelling coordinates.
minor	The minor axis of the elliptical in 2D modelling coordinates.
start	The start angle in radians.
stop	The stop angle in radians.

RETURNS:

None

DESCRIPTION:

When processed by a renderer, this command will cause an elliptical arc to be rendered. The elliptical arc's center and axes are defined in two dimensional modelling coordinate space. The major axis defines the positive X-axis of the ellipse local coordinate system. The minor axis is used in defining the positive Y-axis of the ellipse local coordinate system. A start and end angle relative to the positive X axis define the portion of the circumference drawn to form the specified arc. The arc will lie in the modelling coordinate XY plane (Z = 0).

The arc is drawn in a counter clockwise direction from the starting angle position to the ending angle position in the Z=0 plane.

The major and minor axis parameters define a rectangle which circumscribes the ellipse and is anchored at the center of the ellipse. Two sides are parallel to the major axis. The other two sides are parallel to the minor axis. The major and minor axes extend from the center of the ellipse to midpoints of sides of the parallelogram. The ellipse is tangent to the sides of the parallelogram at their midpoints. The current values of the line attributes are applied to the primitive. The elliptical arc is rendered according to the curve approximation criteria.

ERRORS:

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO:

`PEXSetLineType`, `PEXSetLineWidth`, `PEXSetLineColorIndex`, `PEXSetLineColor`, `PEXSetPolylineInterpMethod`, `PEXSetLineBundleIndex`, `PEXSetCurveApprox`

t
Extended 2D Circular Arc

NAME:

PEXExtCircularArc2D - 2D Circular Arc

SYNTAX:

void PEXExtCircularArc2D(Display **display*, XID *resource_id*,
PEXOCRequest *req_type*, PEXCoord2D **center*, double *radius*,
double *start*, double *stop*)

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`).
center	The center of the circular arc in2D modelling coordinates.
radius	The radius of the circular arc in 2D modelling coordinates.
start	The start angle in radians.
stop	The stop angle in radians.

RETURNS:

None

DESCRIPTION:

When processed by a renderer, this command will cause a circular arc to be rendered. The circular arc's center and radius are defined in two dimensional modelling coordinate space. A center point and a radius specify the circle from which the arc is drawn. A start and end angle relative to the positive X axis define the portion of the circumference drawn to form the specified arc. The arc is drawn in a counter clockwise direction from the starting angle position to the ending angle position in the Z=0 plane.

The current values of the line attributes are applied to the primitive. The circular arc is rendered according to the curve approximation criteria.

If the radius of the circular arc is less than or equal to zero, a dot at the center will be displayed.

ERRORS:

BadPEXRenderer
The specified renderer resource identifier is invalid.

BadPEXStructure
The specified structure resource identifier is invalid.

SEE ALSO:

`PEXSetLineType`, `PEXSetLineWidth`, `PEXSetLineColorIndex`, `PEXSetLineColor`, `PEXSetPolylineInterpMethod`, `PEXSetLineBundleIndex`, `PEXSetCurveApprox`

t Extended Set All Pipeline Context Attributes Value Mask

NAME:

PEXExtSetPCAttributeMaskAll - Macro to Set All Pipeline Context Attributes Value Mask

SYNTAX:

PEXExtSetPCAttributeMaskAll(*mask*)

PARAMETERS:

mask The address of the value mask - an array of three unsigned long.

DESCRIPTION:

This is a utility macro to aid in setting up the bitmask for the extended and the non-extended pipeline context attributes. This macro will set all valid bits in the mask.

ERRORS:

None

SEE ALSO:

`PEXCreatePipelineContext`, `PEXExtChangePipelineContext`, `PEXCopyPipelineContext`, `PEXExtGetPipelineContext`, `PEXExtSetPCAttributeMask`

t Extended Set Pipeline Context Attributes Value Mask

NAME:

PEXExtSetPCAttributeMask - Macro to Setup Extended Pipeline Context Attributes Value Mask

SYNTAX:

PEXExtSetPCAttributeMask(*mask*, *ext_attr*)

PARAMETERS:

mask	The address of the value mask - an array of three unsigned long.
ext_attr	A single pipeline constant attribute bitmask constant or extended pipeline constant attribute bitmask constant.

DESCRIPTION:

This is a utility macro to aid in setting up the bitmask for the extended and the non-extended pipeline context attributes.

The following attribute bitmask constants must be used:

```
PEXPCMarkerType                    <the 5.1 constants>
:
:
PEXPCParaSurfCharacteristics
PEXExtPCTMPerspectiveCorrection    <the extended constants>
PEXExtPCTMResourceHints
PEXExtPCTMSampleFrequency
PEXExtPCActiveTextures
PEXExtPCBFActiveTextures
PEXExtPCPrimitiveAA
PEXExtPCLineCapStyle
PEXExtPCLineJoinStyle
```

Note this macro does multiple evaluations of the value for *ext_attr*.

ERRORS:

None

SEE ALSO:

PEXCreatePipelineContext, PEXExtChangePipelineContext, PEXCopyPipelineContext, PEXExtGetPipelineContext, PEXExtSetPCAttributeMaskAll

t Extended Free Pipeline Context Attributes

NAME:

PEXExtFreePCAttributes - Free Storage Returned by `PEXExtGetPipelineContext`

SYNTAX:

void PEXExtFreePCAttributes(PEXExtPCAttributes **values*)

PARAMETERS:

values A pointer to the extended pipeline context attribute values.

RETURNS:

None

DESCRIPTION:

This function deallocates memory returned by `PEXExtGetPipelineContext`.

ERRORS:

None

SEE ALSO:

`PEXExtGetPipelineContext`

Extended Set All Renderer Attribute Value Mask

NAME:

PEXExtSetRendererAttributeMaskAll - Macro to Set All Extended Renderer Attributes in Value Mask

SYNTAX:

PEXExtSetRendererAttributeMaskAll(*mask*)

PARAMETERS:

mask The address of the value mask - an array of two unsigned long.

DESCRIPTION:

This is a utility macro to aid in setting up the bitmask for the extended and the non-extended renderer attributes. This macro will set all valid bits in the mask.

ERRORS:

None

SEE ALSO:

`PEXExtChangeRenderer`, `PEXExtGetRendererAttributes`

t Extended Set Renderer Attributes Mask

NAME:

PEXExtSetRendererAttributeMask - Macro to Setup Extended Renderer Attributes Value Mask

SYNTAX:

PEXExtSetRendererAttributeMask(*mask*, *ext_attr*)

PARAMETERS:

mask	The address of the value mask - an array of two unsigned long.
ext_attr	A single extended renderer attribute bitmask constant.

DESCRIPTION:

This is a utility macro to aid in setting up the bitmask for the extended and the non-extended renderer attributes.

The following attribute bitmask constants must be used:

```
PEXRAPipelineContext
:
:
PEXRAEchoMode
PEXExtRATMBindingTable
PEXExtRATMCoordSourceTable
PEXExtRATMCompositionTable
PEXExtRATMSamplingTable
```

Note this macro does multiple evaluations of the value for *ext_attr*.

ERRORS:

None

SEE ALSO:

`PEXExtChangeRenderer`, `PEXExtGetRendererAttributes`

t Extended Free Renderer Attributes

NAME:

PEXExtFreeRendererAttributes - Free Storage Returned by `PEXExtGetRendererAttributes`

SYNTAX:

void PEXExtFreeRendererAttributes(PEXExtRendererAttributes **values*)

PARAMETERS:

values A pointer to the extended renderer attribute values.

RETURNS:

None

DESCRIPTION:

This function deallocates memory returned by `PEXExtGetRendererAttributes`.

DATA STRUCTURES:

See also `PEXlib.h` and `PEXExtlib.h`

ERRORS:

None

SEE ALSO:

`PEXExtGetRendererAttributes`

t Extended Compute Fill Area Set With Data Texture Coordinates

NAME:

PEXExtTMCoordFillAreaSetWithData - computes a projection of texture coordinates for a fill area set with data

SYNTAX:

int PEXExtTMCoordFillAreaSetWithData(PEXExtTMCoordData **tm_coord_data*,
unsigned int *vertex_fp_data_size*, unsigned int *facet_attributes*,
unsigned int *vertex_attributes*, int *color_type*,
unsigned int *count*, PEXFacetData **facet_data*,
PEXExtListOfVertex **vertex_lists*)

PARAMETERS:

tm_coord_data	A pointer to parameterization data for this primitive (See *Description* below)
vertex_fp_data_size	The number of floating point data values defined with each vertex.
facet_attributes	A mask indicating the facet attributes provided (`PEXGANone`, `PEXGAColor`, `PEXGANormal`).
vertex_attributes	A mask indicating the vertex attributes provided (`PEXGANone`, `PEXGAColor`, `PEXGANormal`, `PEXGAEdges`, `PEXExtGAData`).
color_type	The type of color data provided (`PEXColorTypeIndexed`, `PEXColorTypeRGB`, `PEXColorTypeCIE`, `PEXColorTypeHSV`, `PEXColorTypeHLS`, `PEXColorTypeRGB8`, `PEXColorTypeRGB16`).
count	The number of fill areas in the set.
facet_data	A pointer to the facet 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:

`PEXExtBadPrimitive`	Normals are not valid.
`PEXExtBadValue`	Projection type is not valid.
`PEXExtBadValue`	Texture coordinate location not valid.
`PEXExtBadAlloc`	Internal allocation error has occurred.

DESCRIPTION:

This function computes a projection of texture coordinates onto a fill area set with data and stores them in the specified vertex data fields.

To compute texture coordinates, the following fields in the *tm_coord_data* structure are initialized:

projection	Texture coordinate projection to perform (`PEXExtTMProjectionSphereWC`, `PEXExtTMProjectionCylinderWC`, `PEXExtTMProjectionLinearWC`)
matrix	If projection is either `PEXExtTMProjectionSphereWC` or `PEXExtTMProjectionCylinderWC`, this transform is applied to vertices and vertex normals after the *mc_transform* has been applied. It is used to orient the data relative to the projection object (sphere, or cylinder) as described.
p0, p1	If projection is `PEXExtTMProjectionLinearWC`, then these elements contain linear projection equations. (See description below.)
coord_source	Source coordinate to compute projections specified with projection type above (`PEXExtTMCoordSourceVertexCoord` or `PEXExtTMCoordSourceVertexNormal`).
fp_data_index	Index within the vertex floating point data list in which to store the calculated texture coordinates. Space must already exist in the vertex floating point data list and the index must point to a valid location.
mc_transform	This transform is applied to vertices and vertex normals in model coordinates prior to computing the specified projection.

Additionally, for the given projections, the following information must be provided. For projections listed below, those which require a unit direction vector (s0,s1,s2) shall have one determined by the coordinate source selection previously mentioned. If `PEXExtTMCoordSourceVertexCoord` is selected, a ray is computed from the WC origin through the vertex to the interior of the projection object. If `PEXExtTMCoordSourceVertexNormal` is selected, the primitive's vertex normals (given or implicit) are used to determine the direction. The resulting values (t0,t1) are the texture coordinates stored at the requested index.

`PEXExtTMProjectionSphereWC`

The infinite sphere has (0,0,0) as its origin and the +Y axis as its axis of revolution. The texture seam sweeps from the +X axis in a counterclockwise direction [0..2PI]. Given the specified coordinate source, a direction vector (s0,s1,s2) is computed to determine a point on the interior of the sphere. The resulting texture coordinates (t0,t1) are stored in the vertex's floating point data list.

```
t0 = (s0 == 0.0) ? ((s2 > 0) ? 0.75 : 0.25) :
    (-arctan(s2/s0) / (2 * PI) +
      ((s0 < 0) ? 0.5 : (s0 > 0 && s2 > 0) ? 1 : 0)
    )

t1 = arcsin(s1) / PI + 0.5

-PI/2 <= arctan(r) <= PI/2
-PI/2 <= arcsin(r) <= PI/2

Where (s0*s0 + s1*s1 + s2*s2) = 1
```

`PEXExtTMProjectionCylinderWC`

The infinite cylinder has (0,0,0) as its origin and the +Y axis as its axis of revolution. The texture seam sweeps from the +X axis in a counterclockwise direction [0..2PI]. Given coordinate sources which are directional (normal or - reflection), a direction vector (s0,s1,s2) is computed to determine a point on the interior of the cylinder. If the coordinate source is vertex coord, a ray perpendicular to the +Y axis through the vertex is computed to determine a point and its height on the interior of the cylinder. The resulting texture coordinates (t0,t1) are stored in the vertex's floating point data list.

```
t0 = (s0 == 0.0) ? ((s2 > 0) ? 0.75 : 0.25) :
    (-arctan(s2/s0) / (2 * PI) +
      ((s0 < 0) ? 0.5 : (s0 > 0 && s2 > 0) ? 1 : 0)
    )

if (coord_source == normal)
    t1 = arcsin(s1) / PI + 0.5
```

else
    t1 = y_coord

-PI/2 <= arctan(r) <= PI/2
-PI/2 <= arcsin(r) <= PI/2

Where (s0*s0 + s1*s1 + s2*s2) = 1

`PEXExtTMProjectionLinearWC`

Parameters for two linear equations are defined a p0 and p1. Given the specified coordinate source, the values will be used directly as (s0,s1,s2) to linearly project the values by the equations listed below. The resulting texture coordinates (t0,t1) are stored in the vertex's floating point data list. (It should be noted that "s3" is assumed to be one as shown in the equations below.)

t0 = s0 * p0[0] + s1 * p0[1] + s2 * p0[2] + 1 * p0[3]
t1 = s0 * p1[0] + s1 * p1[1] + s2 * p1[2] + 1 * p1[3]

It should be noted that the projection of anything other than vertices for this option will lead to unpredictable visual

It is assumed the vertex list of the specified primitive has sufficient space to insert the needed texture coordinates. If space (indicated by `PEXExtGAData`) or valid indices for texture coordinates do not exist, an error will be returned and the data will remain unchanged.

DATA STRUCTURES:

See also `PEXlib.h` and `PEXExtlib.h`

ERRORS:

None

SEE ALSO:

`PEXExtTMCoordSetOfFillAreaSets`, `PEXExtTMCoordTriangleStrip`, `PEXExtTMCoordQuadrilateralMesh`

t Extended Compute Set Of Fill Area Set Texture Coordinates

NAME:

PEXExtTMCoordSetOfFillAreaSets - computes a projection of texture coordinates for a set of fill are sets

SYNTAX:

int PEXExtTMCoordSetOfFillAreaSets(PEXExtTMCoordData **tm_coord_data*,
unsigned int *vertex_fp_data_size*,
unsigned int *facet_attributes*, unsigned int *vertex_attributes*,
int *color_type*, unsigned int *set_count*,
PEXArrayOfFacetData *facet_data*, unsigned int *vertex_count*,
PEXExtArrayOfVertex *vertices*, unsigned int *index_count*,
PEXConnectivityData **connectivity*)

PARAMETERS:

tm_coord_data	A pointer to parameterization data for this primitive (See *Description* in `PEXExtTMCoordFillAreaSetWithData`).
vertex_fp_data_size	The number of floating point data values defined per vertex fill area set attributes.
facet_attributes	A mask indicating the facet attributes provided (`PEXGANone`, `PEXGAColor`, `PEXGANormal`).
vertex_attributes	A mask indicating the vertex attributes provided (`PEXGANone`, `PEXGAColor`, `PEXGANormal`, `PEXExtGAData`).
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)
connectivity	A pointer to the list of contour connectivity data.

RETURNS:

Zero if successful; otherwise, one of the following:

`PEXExtBadPrimitive`	Normals are not valid.
`PEXExtBadValue`	Projection type is not valid.
`PEXExtBadValue`	Texture coordinate location not valid.
`PEXExtBadAlloc`	Internal allocation error has occurred.

DESCRIPTION:

This function computes a projection of texture coordinates onto a set of fill area sets and stores them in the specified vertex data fields.

To compute texture coordinates, the following fields in the *tm_coord_data* structure are initialized:

projection	Texture coordinate projection to perform (`PEXExtTMProjectionSphereWC`, `PEXExtTMProjectionCylinderWC`, `PEXExtTMProjectionLinearWC`)
matrix	If projection is either `PEXExtTMProjectionSphereWC` or `PEXExtTMProjectionCylinderWC`, this transform is applied to vertices and vertex normals after the *mc_transform* has been applied. It is used to orient the data relative to the projection object (sphere, or cylinder) as described.
p0, p1	If projection is `PEXExtTMProjectionLinearWC`, then these elements contain linear projection equations. (See description below.)
coord_source	Source coordinate to compute projections specified with projection type above (`PEXExtTMCoordSourceVertexCoord` or `PEXExtTMCoordSourceVertexNormal`).
fp_data_index	Index within the vertex floating point data list in which to store the calculated texture coordinates. Space must already exist in the vertex floating point data list and the index must point to a valid location.

mc_transform This transform is applied to vertices and vertex normals in model coordinates prior to computing the specified projection.

Additionally, for the given projections, the following information must be provided. For projections listed below, those which require a unit direction vector (s0,s1,s2) shall have one determined by the coordinate source selection previously mentioned. If `PEXExtTMCoordSourceVertexCoord` is selected, a ray is computed from the WC origin through the vertex to the interior of the projection object. If `PEXExtTMCoordSourceVertexNormal` is selected, the primitive's vertex normals (given or implicit) are used to determine the direction. The resulting values (t0,t1) are the texture coordinates stored at the requested index.

`PEXExtTMProjectionSphereWC`

The infinite sphere has (0,0,0) as its origin and the +Y axis as its axis of revolution. The texture seam sweeps from the +X axis in a counterclockwise direction [0..2PI]. Given the specified coordinate source, a direction vector (s0,s1,s2) is computed to determine a point on the interior of the sphere. The resulting texture coordinates (t0,t1) are stored in the vertex's floating point data list.

```
t0 = (s0 == 0.0) ? ((s2 > 0) ? 0.75 : 0.25) :
    (-arctan(s2/s0) / (2 * PI) +
      ((s0 < 0) ? 0.5 : (s0 > 0 && s2 > 0) ? 1 : 0)
    )

t1 = arcsin(s1) / PI + 0.5

-PI/2 <= arctan(r) <= PI/2
-PI/2 <= arcsin(r) <= PI/2

Where (s0*s0 + s1*s1 + s2*s2) = 1
```

`PEXExtTMProjectionCylinderWC`

The infinite cylinder has (0,0,0) as its origin and the +Y axis as its axis of revolution. The texture seam sweeps from the +X axis in a counterclockwise direction [0..2PI]. Given coordinate sources which are directional (normal or - reflection), a direction vector (s0,s1,s2) is computed to determine a point on the interior of the cylinder. If the coordinate source is vertex coord, a ray perpendicular to the +Y axis through the vertex is computed to determine a point and its height on the interior of the cylinder. The resulting texture coordinates (t0,t1) are stored in the vertex's floating point data list.

```
t0 = (s0 == 0.0) ? ((s2 > 0) ? 0.75 : 0.25) :
    (-arctan(s2/s0) / (2 * PI) +
      ((s0 < 0) ? 0.5 : (s0 > 0 && s2 > 0) ? 1 : 0)
```

)

if (coord_source == normal)
    t1 = arcsin(s1) / PI + 0.5
else
    t1 = y_coord

-PI/2 <= arctan(r) <= PI/2
-PI/2 <= arcsin(r) <= PI/2

Where (s0*s0 + s1*s1 + s2*s2) = 1

`PEXExtTMProjectionLinearWC`

Parameters for two linear equations are defined a p0 and p1. Given the specified coordinate source, the values will be used directly as (s0,s1,s2) to linearly project the values by the equations listed below. The resulting texture coordinates (t0,t1) are stored in the vertex's floating point data list. (It should be noted that "s3" is assumed to be one as shown in the equations below.)

t0 = s0 * p0[0] + s1 * p0[1] + s2 * p0[2] + 1 * p0[3]
t1 = s0 * p1[0] + s1 * p1[1] + s2 * p1[2] + 1 * p1[3]

It should be noted that the projection of anything other than vertices for this option will lead to unpredictable visual

It is assumed the vertex list of the specified primitive has sufficient space to insert the needed texture coordinates. If space (indicated by `PEXExtGAData`) or valid indices for texture coordinates do not exist, an error will be returned and the data will remain unchanged.

ERRORS:

None

SEE ALSO:

`PEXExtTMCoordFillAreaSetWithData`, `PEXExtTMCoordTriangleStrip`, `PEXExtTMCoordQuadrilateralMesh`

Extended Compute Triangle Strip Texture Coordinates

NAME:

PEXExtTMCoordTriangleStrip - computes a projection of texture coordinates for a triangle strip

SYNTAX:

int PEXExtTMCoordTriangleStrip(PEXExtTMCoordData **tm_coord_data*,
unsigned int *vertex_fp_data_size*,
unsigned int *facet_attributes*,
unsigned int *vertex_attributes*, int *color_type*,
PEXArrayOfFacetData *facet_data*, unsigned int *count*,
PEXExtArrayOfVertex *vertices*)

PARAMETERS:

tm_coord_data	A pointer to parameterization data for this primitive (See *Description* in `PEXExtTMCoordFillAreaSetWithData`).
vertex_fp_data_size	The number of floating point data values defined per vertex.
facet_attributes	A mask indicating the facet attributes provided (`PEXGANone`, `PEXGAColor`, `PEXGANormal`).
vertex_attributes	A mask indicating the vertex attributes provided (`PEXGANone`, `PEXGAColor`, `PEXGANormal`, `PEXExtGAData`).
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:

Zero if successful; otherwise, one of the following:

`PEXExtBadPrimitive`	Normals are not valid.
`PEXExtBadValue`	Projection type is not valid.

`PEXExtBadValue`	Texture coordinate location not valid.
`PEXExtBadAlloc`	Internal allocation error has occurred.

DESCRIPTION:

This function computes a projection of texture coordinates onto a triangle strip and stores them in the specified vertex data fields.

To compute texture coordinates, the following fields in the *tm_coord_data* structure are initialized:

projection — Texture coordinate projection to perform (`PEXExtTMProjectionSphereWC`, `PEXExtTMProjectionCylinderWC`, `PEXExtTMProjectionLinearWC`)

matrix — If projection is either `PEXExtTMProjectionSphereWC` or `PEXExtTMProjectionCylinderWC`, this transform is applied to vertices and vertex normals after the *mc_transform* has been applied. It is used to orient the data relative to the projection object (sphere, or cylinder) as described.

p0, p1 — If projection is `PEXExtTMProjectionLinearWC`, then these elements contain linear projection equations. (See description below.)

coord_source — Source coordinate to compute projections specified with projection type above (`PEXExtTMCoordSourceVertexCoord` or `PEXExtTMCoordSourceVertexNormal`).

fp_data_index — Index within the vertex floating point data list in which to store the calculated texture coordinates. Space must already exist in the vertex floating point data list and the index must point to a valid location.

mc_transform — This transform is applied to vertices and vertex normals in model coordinates prior to computing the specified projection.

Additionally, for the given projections, the following information must be provided. For projections listed below, those which require a unit direction vector (s0,s1,s2) shall have one determined by the coordinate source selection previously mentioned. If `PEXExtTMCoordSourceVertexCoord` is selected, a ray is computed from the WC origin through the vertex to the interior of the projection object. If `PEXExtTMCoordSourceVertexNormal` is selected, the primitive's vertex normals (given or implicit) are used to determine the direction. The resulting values (t0,t1) are the texture coordinates stored at the requested index.

`PEXExtTMProjectionSphereWC`

The infinite sphere has (0,0,0) as its origin and the +Y axis as its axis of revolution. The texture seam sweeps from the +X axis in a counterclockwise direction [0..2PI]. Given the specified coordinate source, a direction vector (s0,s1,s2) is computed to determine a point on the interior of the sphere. The resulting texture coordinates (t0,t1) are stored in the vertex's floating point data list.

```
t0 = (s0 == 0.0) ? ((s2 > 0) ? 0.75 : 0.25) :
    (-arctan(s2/s0) / (2 * PI) +
        ((s0 < 0) ? 0.5 : (s0 > 0 && s2 > 0) ? 1 : 0)
    )

t1 = arcsin(s1) / PI + 0.5

-PI/2 <= arctan(r) <= PI/2
-PI/2 <= arcsin(r) <= PI/2

Where (s0*s0 + s1*s1 + s2*s2) = 1
```

`PEXExtTMProjectionCylinderWC`

The infinite cylinder has (0,0,0) as its origin and the +Y axis as its axis of revolution. The texture seam sweeps from the +X axis in a counterclockwise direction [0..2PI]. Given coordinate sources which are directional (normal or - reflection), a direction vector (s0,s1,s2) is computed to determine a point on the interior of the cylinder. If the coordinate source is vertex coord, a ray perpendicular to the +Y axis through the vertex is computed to determine a point and its height on the interior of the cylinder. The resulting texture coordinates (t0,t1) are stored in the vertex's floating point data list.

```
t0 = (s0 == 0.0) ? ((s2 > 0) ? 0.75 : 0.25) :
    (-arctan(s2/s0) / (2 * PI) +
        ((s0 < 0) ? 0.5 : (s0 > 0 && s2 > 0) ? 1 : 0)
    )

if (coord_source == normal)
    t1 = arcsin(s1) / PI + 0.5
else
    t1 = y_coord

-PI/2 <= arctan(r) <= PI/2
-PI/2 <= arcsin(r) <= PI/2
```

Where (s0*s0 + s1*s1 + s2*s2) = 1

`PEXExtTMProjectionLinearWC`

Parameters for two linear equations are defined a p0 and p1. Given the specified coordinate source, the values will be used directly as (s0,s1,s2) to linearly project the values by the equations listed below. The resulting texture coordinates (t0,t1) are stored in the vertex's floating point data list. (It should be noted that "s3" is assumed to be one as shown in the equations below.)

t0 = s0 * p0[0] + s1 * p0[1] + s2 * p0[2] + 1 * p0[3]
t1 = s0 * p1[0] + s1 * p1[1] + s2 * p1[2] + 1 * p1[3]

It should be noted that the projection of anything other than vertices for this option will lead to unpredictable visual

It is assumed the vertex list of the specified primitive has sufficient space to insert the needed texture coordinates. If space (indicated by `PEXExtGAData`) or valid indices for texture coordinates do not exist, an error will be returned and the data will remain unchanged.

ERRORS:

None

SEE ALSO:

`PEXExtTMCoordFillAreaSetWithData`, `PEXExtTMCoordSetOfFillAreaSets`, `PEXExtTMCoordQuadrilateralMesh`

t
Extended Compute Quadrilateral Mesh Texture Coordinates

NAME:

PEXExtTMCoordQuadrilateralMesh - computes a projection of texture coordinates for a quadrilateral mesh

SYNTAX:

int PEXExtTMCoordQuadrilateralMesh(PEXExtTMCoordData **tm_coord_data*,
unsigned int *vertex_fp_data_size*,
unsigned int *facet_attributes*,
unsigned int *vertex_attributes*,
int *color_type*, PEXArrayOfFacetData *facet_data*,
unsigned int *col_count*, unsigned int *row_count*,
PEXExtArrayOfVertices *vertices*)

PARAMETERS:

tm_coord_data	A pointer to parameterization data for this primitive (See *Description* in `PEXExtTMCoordFillAreaSetWithData`).
vertex_fp_data_size	The number of floating point data values defined per vertex.
facet_attributes	A mask indicating the facet attributes provided (`PEXGANone`, `PEXGAColor`, `PEXGANormal`).
vertex_attributes	A mask indicating the vertex attributes provided (`PEXGANone`, `PEXGAColor`, `PEXGANormal`, `PEXExtGAData`).
color_type	The type of color data provided (`PEXColorTypeIndexed`, `PEXColorTypeRGB`, `PEXColorTypeCIE`, `PEXColorTypeHSV`, `PEXColorTypeHLS`, `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 quadrilateral mesh.

RETURNS:

Zero if successful; otherwise, one of the following:

`PEXExtBadPrimitive`	Normals are not valid.
`PEXExtBadValue`	Projection type is not valid.
`PEXExtBadValue`	Texture coordinate location not valid.
`PEXExtBadAlloc`	Internal allocation error has occurred.

DESCRIPTION:

This function computes a projection of texture coordinates onto a quadrilateral mesh and stores them in the specified vertex data fields.

To compute texture coordinates, the following fields in the *tm_coord_data* structure are initialized:

projection — Texture coordinate projection to perform (`PEXExtTMProjectionSphereWC`, `PEXExtTMProjectionCylinderWC`, `PEXExtTMProjectionLinearWC`)

matrix — If projection is either `PEXExtTMProjectionSphereWC` or `PEXExtTMProjectionCylinderWC`, this transform is applied to vertices and vertex normals after the *mc_transform* has been applied. It is used to orient the data relative to the projection object (sphere, or cylinder) as described.

p0, p1 — If projection is `PEXExtTMProjectionLinearWC`, then these elements contain linear projection equations. (See description below.)

coord_source — Source coordinate to compute projections specified with projection type above (`PEXExtTMCoordSourceVertexCoord` or `PEXExtTMCoordSourceVertexNormal`).

fp_data_index — Index within the vertex floating point data list in which to store the calculated texture coordinates. Space must already exist in the vertex floating point data list and the index must point to a valid location.

mc_transform — This transform is applied to vertices and vertex normals in model coordinates prior to computing the specified projection.

Additionally, for the given projections, the following information must be provided. For projections listed below, those which require a unit direction vector (s0,s1,s2) shall have one determined by the coordinate source selection previously mentioned. If `PEXExtTMCoordSourceVertexCoord` is selected, a ray is computed from the WC origin through the vertex to the interior of the projection object. If `PEXExtTMCoordSourceVertexNormal` is selected, the primitive's vertex normals (given or implicit) are used to determine the direction. The resulting values (t0,t1) are the texture coordinates stored at the requested index.

`PEXExtTMProjectionSphereWC`

The infinite sphere has (0,0,0) as its origin and the +Y axis as its axis of revolution. The texture seam sweeps from the +X axis in a counterclockwise direction [0..2PI]. Given the specified coordinate source, a direction vector (s0,s1,s2) is computed to determine a point on the interior of the sphere. The resulting texture coordinates (t0,t1) are stored in the vertex's floating point data list.

```
t0 = (s0 == 0.0) ? ((s2 > 0) ? 0.75 : 0.25) :
    (-arctan(s2/s0) / (2 * PI) +
      ((s0 < 0) ? 0.5 : (s0 > 0 && s2 > 0) ? 1 : 0)
    )

t1 = arcsin(s1) / PI + 0.5

-PI/2 <= arctan(r) <= PI/2
-PI/2 <= arcsin(r) <= PI/2

Where (s0*s0 + s1*s1 + s2*s2) = 1
```

`PEXExtTMProjectionCylinderWC`

The infinite cylinder has (0,0,0) as its origin and the +Y axis as its axis of revolution. The texture seam sweeps from the +X axis in a counterclockwise direction [0..2PI]. Given coordinate sources which are directional (normal or - reflection), a direction vector (s0,s1,s2) is computed to determine a point on the interior of the cylinder. If the coordinate source is vertex coord, a ray perpendicular to the +Y axis through the vertex is computed to determine a point and its height on the interior of the cylinder. The resulting texture coordinates (t0,t1) are stored in the vertex's floating point data list.

```
t0 = (s0 == 0.0) ? ((s2 > 0) ? 0.75 : 0.25) :
    (-arctan(s2/s0) / (2 * PI) +
      ((s0 < 0) ? 0.5 : (s0 > 0 && s2 > 0) ? 1 : 0)
    )

if (coord_source == normal)
    t1 = arcsin(s1) / PI + 0.5
```

else
    t1 = y_coord

-PI/2 <= arctan(r) <= PI/2
-PI/2 <= arcsin(r) <= PI/2

Where (s0*s0 + s1*s1 + s2*s2) = 1

`PEXExtTMProjectionLinearWC`

Parameters for two linear equations are defined as p0 and p1. Given the specified coordinate source, the values will be used directly as (s0,s1,s2) to linearly project the values by the equations listed below. The resulting texture coordinates (t0,t1) are stored in the vertex's floating point data list. (It should be noted that "s3" is assumed to be one as shown in the equations below.)

t0 = s0 * p0[0] + s1 * p0[1] + s2 * p0[2] + 1 * p0[3]
t1 = s0 * p1[0] + s1 * p1[1] + s2 * p1[2] + 1 * p1[3]

It should be noted that the projection of anything other than vertices for this option will lead to unpredictable visual

It is assumed the vertex list of the specified primitive has sufficient space to insert the needed texture coordinates. If space (indicated by `PEXExtGAData`) or valid indices for texture coordinates do not exist, an error will be returned and the data will remain unchanged.

ERRORS:

None

SEE ALSO:

`PEXExtTMCoordFillAreaSetWithData`, `PEXExtTMCoordSetOfFillAreaSets`, `PEXExtTMCoordTriangleStrip`

t Extended Create Filtered Texture Map

NAME:

PEXExtCreateFilteredTM - creates a prefiltered texture map from base map

SYNTAX:

int PEXExtCreateFilteredTM(int *domain*, PEXExtTMDomainData **domain_data*,
unsigned int *power_of_two_tm_required*,
unsigned int *square_tm_required*.
PEXExtTexelArray **base_map*,
PEXExtTexelArray ***texel_array*)

PARAMETERS:

domain	The texture map domain type.
domain_data	A pointer to domain information specifying the type of texture map to be created and the type of texel data.
power_of_two_tm_required	Indicates whether maps with dimensions of power of two is required (True, False).
square_tm_required	Indicates whether square maps is required (True, False).
base_map	The texel data of the desired filtered texture map's base level.
texel_array	Texel data created by this utility.

RETURNS:

Zero if successful; otherwise, one of the following:

`PEXExtBadDomain`	Domain is not valid.
`PEXExtBadDimension`	Dimension is not valid.
`PEXExtBadTexelType`	Texel type is not valid.
`PEXExtBadParameterization`	Parameterization type is not valid.
`PEXExtBadTMAlloc`	Memory allocation error.
`PEXExtBadTexelDimension`	Texel dimensions are not valid.
`PEXExtBadTexelData`	Texel data is not valid.

DESCRIPTION:

This function creates a prefiltered texture map and returns a list of texel arrays suitable for `PEXExtCreateTM`. The information contained in *domain* and *domain_data* will determine if a MipMap or other type of texture map is created from the base level data provided.

A value of zero in the domain data field *num_levels* indicates to the routine that it should generate as many levels as necessary to create a full MipMap pyramid or other type of supported map. If the value is non-zero, then the utility will create either the number of levels requested or all the levels for a full map, which ever is less. Upon exit, the *num_levels* field will be updated to the number of levels actually created by the utility.

Implementation constraints will be observed in the creation of the final texel data. If an implementation requires that maps be created with square dimensions and/or dimensions which are a power of two, the utility will apply the appropriate image sizing to the nearest correct dimension. When shrinking the image, the texels from the source image is sampled down using a box filter to create the destination image. When magnifying the image, the texels from the source image will be linearly interpolated to create the destination image.

Texel arrays allocated by this function are disposed of using the `PEXExtFreeFilteredTM` utility after they have been used by the `PEXExtCreateTM` function.

DATA STRUCTURES:

See also `PEXlib.h` and `PEXExtlib.h`

ERRORS:

None

SEE ALSO:

`PEXExtCreateTM`, `PEXExtFreeFilteredTM`

t Extended Create Filtered Texture Map from Window

NAME:

PEXExtCreateFilteredTMFromWindow - creates a prefiltered texture map from a designated window

SYNTAX:

int PEXExtCreateFilteredTMFromWindow(Display **display*,
int *domain*, PEXExtTMDomainData **domain_data*,
unsigned int *power_of_two_tm_required*,
unsigned int *square_tm_required*,
unsigned int *luminance_channel_selector*,
XID *base_color_window*,
unsigned int *alpha_channel_selector*,
XID *base_alpha_window*,
PEXExtTexelArray ***texel_array*)

PARAMETERS:

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

domain — The texture map domain type.

domain_data — A pointer to domain information specifying the type of texture map to be created and the type of texel data.

power_of_two_tm_required — Indicates whether maps with dimensions of power of two is required (True, False).

square_tm_required — Indicates whether square maps is required (True, False).

luminance_channel_selector — Luminance channel source selector. (`PEXExtChannelNTSCLuminance`, `PEXExtChannelRed`, `PEXExtChannelGreen`, `PEXExtChannelBlue`)

base_color_window — The X window identifier of the desired filtered texture map's base level color and/or luminance data.

alpha_channel_selector
: Alpha channel source selector from alpha window, if relevant to texel type to be created. (`PEXExtChannelNTSCLuminance`, `PEXExtChannelRed`, `PEXExtChannelGreen`, `PEXExtChannelBlue`)

base_alpha_window
: The X window identifier of the desired filtered texture map's base level alpha data.

texel_array
: Texel data created by this utility.

RETURNS:

Zero if successful; otherwise, one of the following:

`PEXExtBadXResource`	X resource ID is not valid.
`PEXExtBadDomain`	Domain is not valid.
`PEXExtBadTexelType`	Texel type is not valid.
`PEXExtBadTMAlloc`	Memory allocation error.
`PEXExtBadTMType`	Texture map type is not valid.
`PEXExtBadValue`	Luminance channel selector is not valid.
`PEXExtBadValue`	Alpha channel selector is not valid.

DESCRIPTION:

This function creates a prefiltered texture map and returns texel data suitable for `PEXExtCreateTM`. The information contained in *domain* and *domain_data* will determine if a MipMap or other type of texture map is created from the base level data provided.

The *base_color_window* parameter identifies a fully exposed and unobscured window from which to read texel data. If the texel type to create indicates alpha, then *base_alpha_window* identifies a fully exposed and unobscured window from which to read alpha texel data. If either window is obscured, then the results of the output texel array may be undefined.

A value of zero in the domain data field *num_levels* indicates to the routine that it should generate as many levels as necessary to create a full MipMap pyramid or other type of supported map. If the value is non-zero, then the utility will create either the number of levels requested or all the levels for a full map, whichever is less. Upon exit, the *num_levels* field will be updated to the number of levels actually created by the utility.

If the texel type specified in the description indicates luminance is included (i.e. the texel type is `PEXExtTexelLuminanceFloat`, `PEXExtTexel-LuminanceInt8`, `PEXExtTexelLuminanceInt16`, `PEXExtTexelLuminanceAlphaFloat`, `PEXExtTexelLuminanceAlphaInt8`, or `PEXExtTexelLuminanceAlphaInt16`), the *luminance_channel_selector* indicates the source of the luminance data. The channels available as source of the luminance data are dependent upon the depth or visual class of *base_color_window*.

If the texel type specified in the description indicates alpha is included (i.e. the texel type is `PEXExtTexelLuminanceAlphaFloat`, `PEXExtTexelLuminanceAlphaInt8`, `PEXExtTexelLuminanceAlphaInt16`, `PEXExtTexelRGBAlphaFloat`, `PEXExtTexelRGBAlphaInt8`, or `PEXExtTexelRGBAlphaInt16`), the *alpha_channel_selector* indicates the source of the alpha data. The channels available as source of the alpha data are dependent upon the depth or visual class of *base_alpha_window*.

Selecting `PEXExtChannelNTSCLuminance` results in calculating the luminance intensities from the active color map of the window resource as:

luminance = (red * 0.299 + green * 0.587 + blue * 0.114)

Selecting any other channel type directly uses the data in that channel as a source for luminance or alpha data.

Implementation constraints will be observed in the creation of the final texel data as indicated by *power_of_two_tm_required* and *square_tm_required*. If an implementation requires that maps be created with square dimensions and/or dimensions which are a power of two, the utility will apply the appropriate image sizing to the nearest correct dimension. When shrinking the image, the texels from the source image are sampled down using a box filter to create the destination image. When magnifying the image, the texels from the source image will be linearly interpolated to create the destination image.

Texel arrays allocated by this function are disposed of using the `PEXExtFreeFilteredTM` utility after they have been used by the `PEXExtCreateTM` function.

DATA STRUCTURES:

See also `PEXlib.h` and `PEXExtlib.h`

ERRORS:

None

SEE ALSO:

`PEXExtCreateTM`, `PEXExtFreeFilteredTM`

t Extended Free Filtered Texture Map

NAME:

PEXExtFreeFilteredTM - frees up texture resources

SYNTAX:

void PEXExtFreeFilteredTM(int *domain*, PEXExtTMDomainData **domain_data*,
PEXExtTexelArray **texel_array*)

PARAMETERS:

domain	The texture map domain type.
domain_data	A pointer to texture map domain information.
texel_array	Texel data to be freed.

RETURNS:

None

DESCRIPTION:

This function frees storage returned by `CreateFilteredTM` utilities. Texel data created by these functions can be immediately freed by `PEXExtFreeFilteredTM` after using them with the `PEXExtCreateTM` function.

DATA STRUCTURES:

See also `PEXlib.h` and `PEXExtlib.h`

ERRORS:

None

SEE ALSO:

`PEXExtCreateTM, PEXExtCreateFilteredTM`
`PEXExtCreateFilteredTMFromWindow`

t

Extended Count Output Commands

NAME:

PEXExtCountOCs - Extended Count Output Commands

SYNTAX:

unsigned long PEXExtCountOCs(int *float_format*, 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)
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.

DATA STRUCTURES:

See also `PEXlib.h` and `PEXExtlib.h`

ERRORS:

None

SEE ALSO:

`PEXExtDecodeOCs`, `PEXExtEncodeOCs`, `PEXExtFreeOCData`

Extended Decode Output Commands

NAME:

PEXExtDecodeOCs - Extended Decode Output Commands

SYNTAX:

PEXExtOCData *PEXExtDecodeOCs(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 extended 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. PEXExtFreeOCData 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.

DATA STRUCTURES:

See also `PEXlib.h` and `PEXExtlib.h`

ERRORS:

None

SEE ALSO:

`PEXExtEncodeOCs, PEXExtFreeOCData`

t Extended Encode Output Commands

NAME:

PEXExtEncodeOCs - Extended Encode Output Commands

SYNTAX:

char *PEXExtEncodeOCs(int *float_format*, unsigned long *oc_count*,
PEXExtOCData **oc_data*, unsigned long **length_return*)

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 to be encoded.
oc_data	An array of the extended 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. XFree should be called to deallocated 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.

DATA STRUCTURES:

See also `PEXlib.h` and `PEXExtlib.h`

ERRORS:

None

SEE ALSO:

`PEXExtDecodeOCs`

Extended Free Output Commands Data

NAME:

PEXExtFreeOCData - Extended Deallocate OC Data

SYNTAX:

void PEXExtFreeOCData(unsigned long *count*, PEXExtOCData **oc_data*)

PARAMETERS:

count	The number of output commands.
oc_data	An array of the extended output command data.

RETURNS:

None

DESCRIPTION:

This function has no visible effect. This function deallocates memory allocated by `PEXExt-DecodeOCs` to hold decoded output command data.

DATA STRUCTURES:

See also `PEXlib.h` and `PEXExtlib.h`

ERRORS:

None

SEE ALSO:

`PEXExtDecodeOCs`

t

Extended Get Output Commands Size

NAME:

PEXExtGetSizeOCs - Extended Get OC Size

SYNTAX:

int PEXExtGetSizeOCs (int *float_format*, int *oc_count*, PEXExtOCData **oc_data*)

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.
oc_data	An array of the extended 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 extended output commands. An successful return value is possible if the specified floating point format is invalid or unsupported by PEXlib.

DATA STRUCTURES:

See also `PEXlib.h` and `PEXExtlib.h`

ERRORS:

None

t **Extended Fetch Elements and Send**

NAME:

PEXExtFetchElementsAndSend - Extended Fetch Elements and Send to Display

SYNTAX:

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

PARAMETERS:

src_display	A pointer to a display structure returned by a successful `XOpenDisplay` call.
structure	The resource identifier of the structure.
whence1	A value of 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 `PEXExtFetchElements` 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 `PEXExtFetchElements`, and then sending the returned list of output commands by calling `PEXStartOC`, `PEXCopyBytesToOC` and `PEXFinishOC`.

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, 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 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.

DATA STRUCTURES:

See also `PEXlib.h` and `PEXExtlib.h`

ERRORS:

BadPEXStructure
The specified structure resource identifier is invalid.

BadValue
A specified value for *whence* parameter is invalid.

SEE ALSO

`PEXExtFetchElements`