

**NAME**

XeviQueryExtension, XeviQueryVersion, XeviGetVisualInfo - X Extended Visual Information functions

**SYNOPSIS**

```
#include <X11/extensions/XEVI.h>
```

```
Bool XeviQueryExtension (Display *dpy);
```

```
Bool XeviQueryVersion (Display *dpy,
    int *major_version_return,
    int *minor_version_return);
```

```
int XeviGetVisualInfo (Display *dpy, VisualID *visual,
    int n_visual, ExtendedVisualInfo ** evi_return,
    int * n_info_return);
```

**DESCRIPTION**

The X11 Extended Visual Information extension (EVI) allows a client to determine information about core X visuals beyond what the core protocol provides.

The EVI application programming library contains the interfaces described below. With the exception of **XeviQueryExtension**, if any of these routines are called with a display that does not support the extension, the **ExtensionErrorHandler** (which can be set with **XSetExtensionErrorHandler** and functions the same way as **XSetErrorHandler**) will be called and the function will then return.

**XeviQueryExtension** returns **True** if the Extended Visual Information extension is available on the given display. A client must call **XeviQueryExtension** before calling any other EVI function in order to negotiate a compatible protocol version; otherwise the client will get undefined behavior (EVI may or may not work).

**XeviQueryVersion** returns **True** if the request succeeded; the values of the major and minor protocol version supported by the server are returned in *major\_version\_return* and *minor\_version\_return*.

**XeviGetVisualInfo** returns a list of **ExtendedVisualInfo** structures that describe visual information beyond that supported by the core protocol. This includes layer information relevant for systems supporting overlays and/or underlay planes, and information that allows applications better to determine the level of hardware support for multiple colormaps. **XeviGetVisualInfo** returns **Success** if successful, or an X error otherwise. If the argument *visual* is **NULL**, then information for all visuals of all screens is returned. Otherwise, it's a pointer to a list of visuals for which extended visual

information is desired. *n\_visual* is the number of elements in the array *visual*. *evi\_return* returns a pointer to a list of `ExtendedVisualInfo`. When done, the client should free the list using `XFree`. *n\_info\_return* returns the number of elements in the array *evi\_return*.

The **ExtendedVisualInfo** structure has the following fields:

VisualID	<i>core_visual_id</i>
int	<i>screen</i>
int	<i>level</i>
unsigned int	<i>transparency_type</i>
unsigned int	<i>transparency_value</i>
unsigned int	<i>min_hw_colormaps</i>
unsigned int	<i>max_hw_colormaps</i>
unsigned int	<i>num_colormap_conflicts</i>
VisualID *	<i>colormap_conflicts</i>

The combination of *core\_visual\_id* and *screen* number uniquely specify the visual being described.

*level* returns the level number for the visual, 0 for normal planes, > 0 for overlays, < 0 for underlays.

*transparency\_type* returns the type of transparency supported by the visual.

`XEVI_TRANSPARENCY_NONE` if there are no transparent pixels,

`XEVI_TRANSPARENCY_PIXEL` if the visual supports a transparent pixel,

`XEVI_TRANSPARENCY_MASK` if the visual supports transparent plane(s).

*transparency\_value* returns the pixel/plane value to set for transparency if *transparency\_type* isn't `XEVI_TRANSPARENCY_NONE`.

*min\_hw\_colormaps* and *max\_hw\_colormaps* return the minimum and maximum number of hardware colormaps backing up the visual.

*num\_colormap\_conflicts* returns the number of elements in *colormap\_conflicts*. This array returns a list of visuals that may cause conflicts in the use of the hardware colormap. For example, if a 12-bit hardware colormap is overloaded to support 8-bit colormaps, the corresponding 8-bit visuals would conflict with the 12-bit visuals.

## ERRORS

**XeviGetVisualInfo** will return *BadValue* if passed an illegal visual ID, *BadAccess* if the X server does not respond, *BadAlloc* if there is a memory allocation failure.