NAME

XIQueryDevice, XIFreeDeviceInfo - get information about devices.

SYNOPSIS

```
#include <X11/extensions/XInput2.h>

XIDeviceInfo* XIQueryDevice( Display *display,
                             int deviceid,
                             int *ndevices_return);

XIFreeDeviceInfo( XIDeviceInfo *info);
```

deviceid
    Specifies the device to query or XIAllDevices or XIAllMasterDevices.

display
    Specifies the connection to the X server.

ndevices_return
    Returns the number of devices returned.

info
    A list of device XIDeviceInfo structs to be freed.

DESCRIPTION

The XIQueryDevice returns information about one or more input devices. If the deviceid specifies a device, ndevices_return is 1 and the returned information describes only the requested device. If deviceid is XIAllDevices or XIAllMasterDevices, ndevices_return is the number of devices or master devices, respectively, and the returned information represents all devices or all master devices, respectively.

To free the XIDeviceInfo array returned by XIQueryDevice, use XIFreeDeviceInfo.

For each input device requested, the XIQueryDevice returns an XIDeviceInfo structure. Each structure contains information about the capabilities of one input device available to the server.

```
typedef struct
{
    int                 deviceid;
    char                *name;
    int                 use;
    int                 attachment;
    Bool                enabled;
    int                 num_classes;
    XIAnyClassInfo      **classes;
} XIDeviceInfo;
```

The deviceid is the numeric unique id of the device. A deviceid is unique for the life-time of a device but a server may re-use the id once a device has been removed.

The name points to a null-terminated string specifying the identifier of the device.

The use and attachment fields specify the type of the device and the current attachment or pairing.
- If use is XIMasterPointer, the device is a master pointer and attachment specifies the deviceid of the paired master keyboard.
- If use is XIMasterKeyboard, the device is a master keyboard, and the attachment field specifies the paired master pointer.
- If use is XISlavePointer, the device is a slave device and currently attached to the master pointer specified in attachment.
- If use is XISlaveKeyboard, the device is a slave device an currently attached to the master keyboard specified in attachment.
- If use is XIFloatingSlave, the device is a slave device currently not attached to any master device. The value of the attachment field for floating slave devices is undefined.

The enabled field specifies if the device is currently enabled and can send events. Disabled devices will not send events.

The num_classes field specifies the number of input classes pointed to by classes. The first two fields of all input classes are identical.

```
typedef struct
{
   int       type;
   int       sourceid;
} XIAnyClassInfo;
```

The type field specifies the type of the input class. Currently, the following types are defined:
   XIKeyClass, XIButtonClass, XIValuatorClass, XIScrollClass,
   XITouchClass, XIGestureClass.

In the future, additional types may be added. Clients are required to ignore unknown input classes.

The sourceid is the deviceid this class originated from. For master devices, the sourceid is typically the id of the slave device currently sending events. For slave devices, the sourceid is typically the device's id.

A device may have zero or one XIButtonClass, denoting the device's capability to send button events.

```
typedef struct {
   int         mask_len;
   unsigned char *mask;
} XIButtonState;

typedef struct
{
   int       type;
   int       sourceid;
   int       num_buttons;
   Atom       *labels;
   XIButtonState state;
} XIButtonClassInfo;
```

The num_buttons field specifies the number of buttons available on this device. A device that has an XIButtonClass must have at least one button.

labels is a list of num_buttons Atoms specifying the button labels for this device. If the label is not None, then the label specifies the type of button in physical device order (i.e. as the buttons are numbered on the physical input device).

The state is the current button state as seen by clients (i.e. after button mapping is applied). The mask_len field specifies the length of mask in bytes. For each button on the device, the respective bit in mask is set if the button is currently logically down.

A device may have zero or one XIKeyClass, denoting the device’s capability to send key events.

```
typedef struct
{
    int         type;
    int         sourceid;
    int         num_keycodes;
    int         *keycodes;
} XIKeyClassInfo;
```

The num_keycodes field specifies the number of keycodes available on this device. A device that has an XIKeyClass must have at least one keycode.

keycodes is a list of num_keycodes keycodes the device may send.

A device may have zero or more XIValuatorClass, denoting the device’s capability to send coordinates.

```
typedef struct
{
    int         type;
    int        sourceid;
    int        number;
    Atom       label;
    double     min;
    double     max;
    double     value;
    int        resolution;
    int        mode;
} XIValuatorClassInfo;
```

The number field specifies the number of the axis on the physical device.

If the label field is not None, the value of label is an Atom describing the axis.

min and max are the minimum and maximum values allowed on this axis. If both are zero, no minimum or maximum values are set on this device. value is the current value of this axis.

The resolution field specifies the resolution of the device in units/m.

The mode specifies the mode of this axis. If the mode is XIModeAbsolute this axis sends absolute coordinates. If the mode is XIModeRelative, this device sends relative coordinates.

```
typedef struct
{
    int        type;
    int        sourceid;
    int        number;
    int        scroll_type;
    double     increment;
    int        flags;
} XIScrollClassInfo;
```

This class describes scrolling capability on a valuator. For each XIScrollClassInfo, an XIValuatorClassInfo with the same number is present on the device.

The number field specifies the valuator number on the physical
device that this scroll information applies to. See the
respective XIValuatorClassInfo for detailed information on this
valuator.

The scroll_type field specifies the type of scrolling, either
XIScrollTypeVertical or XIScrollTypeHorizontal.

The increment specifies the value change considered one unit of
scrolling down.

The flags field specifies flags that apply to this scrolling
information:

If XIScrollFlagNoEmulation is set, the server will not
emulate legacy button events for valuator changes on this
valuator.

If XIScrollFlagPreferred is set, this axis is the
preferred axis for this scroll type and will be used for
the emulation of XI_Motion events when the driver submits
legacy scroll button events.

```
typedef struct
{
   int       type;
   int       sourceid;
   int       mode;
   int       num_touches;
} XITouchClassInfo;
```

A device may have zero or one XITouchClassInfo, denoting
multi-touch capability on the device. A device with a XITouchClassInfo
may send TouchBegin, TouchUpdate, TouchEnd and TouchOwnership events.

The mode field is either XIDirectTouch for direct-input touch devices
such as touchscreens or XIDependentTouch for indirect input devices such
as touchpads. For XIDirectTouch devices, touch events are sent to window
at the position the touch occurred. For XIDependentTouch devices, touch
events are sent to the window at the position of the device’s sprite.

The num_touches field defines the maximum number of simultaneous touches the device supports. A num_touches of 0 means the maximum number of simultaneous touches is undefined or unspecified. This field should be used as a guide only, devices will lie about their capabilities.

A device with an XITouchClassInfo may still send pointer events. The valuators must be defined with the respective XIValuatorClass classes. A valuator may send both pointer and touch-events.

```
typedef struct
{
    int         type;
    int         sourceid;
    int         num_touches;
} XIGestureClassInfo;
```

A device may have zero or one XIGestureClassInfo, denoting touchpad gesture capability on the device. A device with a XIGestureClassInfo may send GestureSwipeBegin, GestureSwipeUpdate, GestureSwipeEnd, GesturePinchBegin, GesturePinchUpdate, GesturePinchEnd.

The num_touches field defines the maximum number of simultaneous touches the device supports. A num_touches of 0 means the maximum number of simultaneous touches is undefined or unspecified. This field should be used as a guide only, devices will lie about their capabilities.

XIQueryDevice can generate a BadDevice error.

XIFreeDeviceInfo frees the information returned by XIQueryDevice.

DIAGNOSTICS

BadDevice

An invalid device was specified. The device does not exist or is not a pointer device.