XOpenDevice, XCloseDevice – open or close an extension input device

XDevice *XOpenDevice(*display***,** *device_id***)**
Display ***display*;**
XID *device_id***;**

XCloseDevice(*display***,** *device***)**
Display ***display*;**
XDevice ***device*;**

display **Specifies the connection to the X server.** *device_id* **Specifies the id of the device to be opened** *device* **Specifies the device to be closed**

The *XOpenDevice* **request makes an input device accessible to a client through input extension protocol requests. If successful, it returns a pointer to an XDevice structure.**

The *XCloseDevice* request makes an input device inaccessible to a client through input extension protocol requests. Before terminating, and client that has opened input devices through the input extension should close them via *CloseDevice*.

When a client makes an *XCloseDevice* request, any active grabs that the client has on the device are released. Any event selections that the client has are deleted, as well as any passive grabs. If the requesting client is the last client accessing the device, the server may disable all access by X to the device.

XOpenDevice and *XCloseDevice* can generate a *BadDevice* error.

The *XDevice* **structure returned by** *XOpenDevice* **contains:**

```
typedef struct {
        XID device_id;
        int num_classes;
        XInputClassInfo *classes;
} XDevice;
```

The classes field is a pointer to an array of XInputClassInfo structures. Each element of this array contains an event type base for a class of input supported by the specified device. The num_classes field indicates the number of elements in the classes array.

The *XInputClassInfo* structure contains:

```
typedef struct {
        unsigned char input_class;
        unsigned char event_type_base;
} XInputClassInfo;
```

The input_class field identifies one class of input supported by the device. Defined types include *KeyClass*, *ButtonClass*, *ValuatorClass*, *ProximityClass*, *FeedbackClass*, *FocusClass*, and *OtherClass.* The event_type_base identifies the event type of the first event in that class.

The information contained in the *XInputClassInfo* structure is used by macros to obtain the event classes that clients use in making *XSelectExtensionEvent* requests. Currently defined macros include *DeviceKeyPress*, *DeviceKeyRelease*, *DeviceButtonPress*, *DeviceButtonRelese*, *DeviceMotionNotify*, *DeviceFocusIn*, *DeviceFocusOut*, *ProximityIn*, *ProximityOut*, *DeviceStateNotify*, *DeviceMappiingNotify*, *ChangeDeviceNotify*, *DevicePointerMotionHint*, *DeviceButton1Motion*, *DeviceButton2Motion*, *DeviceButton3Motion*, *DeviceButton4Motion*, *DeviceButton5Motion*, *DeviceButtonMotion*, *DeviceOwnerGrabButton*, *DeviceButtonPressGrab*, and *NoExtensionEvent.*

To obtain the proper event class for a particular device, one of the above macros is invoked using the *XDevice* structure for that device. For example,

DeviceKeyPress (*device, type, eventclass);

returns the *DeviceKeyPress* event type and the eventclass for *DeviceKeyPress* events from the specified device.

This *eventclass* can then be used in an *XSelectExtensionEvent* request to ask the server to send *DeviceKeyPress* events from this device. When a selected event is received via *XNextEvent*, the *type* can be used for comparison with the type in the event.

BadDevice **An invalid device was specified. The specified device does not exist, or is the X keyboard or X pointer. This error may also occur if some other client has caused the specified device to become the X keyboard or X pointer device via the** *XChangeKeyboardDevice* **or** *XChangePointerDevice* **requests.**