NAME

xinput - utility to configure and test X input devices

SYNOPSIS

xinput [OPTIONS] [DEVICE]

DESCRIPTION

xinput is a utility to list available input devices, query information about a device and change input device settings.

OPTIONS

--version Test if the X Input extension is available and return the version number of the program and the version supported by the server. This option does not require a device name.

--list [--short || --long || --name-only || --id-only] [*device***]**

If no argument is given list all the input devices. If an argument is given, show all the features of *device*. If --long is provided, the output includes detailed information about the capabilities of each devices. Otherwise, or if --short is provided, only the device names and some minimal information is listed. If --name-only is provided, the output is limited to the device names. One device name is listed per line. Note that the order the devices are listed is undefined. If --id-only is provided, the output is limited to the device IDs. One device ID is listed per line. Note that the order the devices are listed is undefined.

--get-feedbacks *device*

Display the feedbacks of *device*.

--set-pointer *device*

Switch *device* in core pointer. This option does nothing on X servers 1.5 and later.

--set-mode *device ABSOLUTE|RELATIVE*

Change the mode of *device*.

--set-ptr-feedback *device threshold num denom*

Change the pointer acceleration (or feedback) parameters of *device*. The xset(1) man page has more details. For X.Org Server 1.7 and above, there are additional device properties pertaining to pointer acceleration. These do not replace, but complement the pointer feedback setting.

--set-integer-feedback *device index value*
Change the value of an integer feedback of *device*.

--set-button-map *device map_button_1* **[***map_button_2* **[...]]**
Change the button mapping of *device*. The buttons are specified in physical order (starting with button 1) and are mapped to the logical button provided. 0 disables a button. The default button mapping for a device is 1 2 3 4 5 6 etc.

--query-state *device*
Query the device state.

--list-props *device* **[***device* **[...]]**
Lists properties that can be set for the given device(s).

--set-int-prop *device property format value*
Sets an integer property for the device. Appropriate values for *format* are 8, 16, or 32, depending on the property. Deprecated, use **--set-prop** instead.

--set-float-prop *device property value*
Sets a float property for the device. Deprecated, use **--set-prop** instead.

--set-prop [--type=*atom|float|int***] [--format=***8|16|32***]** *device property value* **[...]**
Set the property to the given value(s). If not specified, the format and type of the property are left as-is. The arguments are interpreted according to the property type. See Section *CHANGING PROPERTIES*.

--watch-props *device*
Prints to standard out when property changes occur.

--delete-prop *device property*
Delete the property from the device.

--test [-proximity] *device*
Register all extended events from *device* and enter an endless loop displaying events received. If the -proximity is given, ProximityIn and ProximityOut are registered.

--test-xi2 [--root] [*device***]**
Register for a number of XI2 events and display them. If a device is given, only events on this device are displayed. If --root is given, events are selected on the root window only. Otherwise, a new client window is created (similar to xev).

--create-master *prefix* **[sendCore] [enable]**

Create a new pair of master devices on an XI2-enabled server with the given *prefix*. The server will create one master pointer named "*prefix* pointer" and one master keyboard named "*prefix* keyboard". If *sendCore* is 1, this pair of master devices is set to send core events (default). If *enable* is 1, this master device pair will be enabled immediately (default).

--remove-master *master* **[Floating|AttachToMaster] [returnPointer] [returnKeyboard]**

Remove *master* and its paired master device. Attached slave devices are set floating if *Floating* is specified or the argument is omitted. If the second argument is *AttachToMaster*, *returnPointer* specifies the master pointer to attach all slave pointers to and *returnKeyboard* specifies the master keyboard to attach all slave keyboards to.

--reattach *slave master*

Reattach *slave* to *master*.

--float *slave*

Remove *slave* from its current master device.

--set-cp *window master*

Set the ClientPointer for the client owning *window* to *master*. *master* must specify a master pointer.

--map-to-output *device crtc*

Restricts the movements of the absolute *device* to the RandR *crtc*. The output name must match a currently connected output (see *xrandr(1)*). If the NVIDIA binary driver is detected or RandR 1.2 or later is not available, a Xinerama output may be specified as "HEAD-N", with N being the Xinerama screen number. This option has no effect on relative devices.

--enable *device*

Enable the *device*. This call is equivalent to **xinput --set-prop device** *"Device Enabled"* **1**

--disable *device*

Disable the *device*. This call is equivalent to **xinput --set-prop device** *"Device Enabled"* **0**

device can be the device name as a string or the XID of the device.

slave can be the device name as a string or the XID of a slave device.

master can be the device name as a string or the XID of a master device.

property can be the property as a string or the Atom value.

CHANGING PROPERTIES

When xinput should modify an existing driver property value, it is sufficient to provide the device name and property name as string, followed by the new value(s) of the property. For example:

xinput set-prop "my device" "my prop" 1 2 3

XWAYLAND

Xwayland is an X server that uses a Wayland Compositor as backend. Xwayland acts as translation layer between the X protocol and the Wayland protocol but does not have direct access to the hardware. The X Input Extension devices created by Xwayland ("xwayland-pointer", "xwayland-keyboard", etc.) map to the Wayland protocol devices, not to physical devices.

These X Input Extension devices are only visible to other X clients connected to the same Xwayland process. Changing properties on Xwayland devices only affects the behavior of those clients. For example, disabling an Xwayland device with xinput does not disable the device in Wayland-native applications. Other changes may not have any effect at all.

In most instances, using xinput with an Xwayland device is indicative of a bug in a shell script and xinput will print a warning. Use the Wayland Compositor’s native device configuration methods instead.

SEE ALSO

X(7), xset(1), xrandr(1)

COPYRIGHT

AUTHORS

Peter Hutterer <peter.hutterer@who-t.net>

Philip Langdale, <philipl@alumni.utexas.net>
Frederic Lepied, France <Frederic.Lepied@sugix.frmug.org>
Julien Cristau <jcristau@debian.org>
Thomas Jaeger <ThJaeger@gmail.com>
and more.