NAME

XkbGetVirtualMods - Obtain a subset of the virtual modifier bindings (the vmods array) in a keyboard description

SYNOPSIS

Status XkbGetVirtualMods (Display **dpy*, unsigned int *which*, XkbDescPtr *xkb*);

ARGUMENTS

dpy connection to server

which
mask indicating virtual modifier bindings to get

xkb Xkb description where results will be placed

DESCRIPTION

XkbGetVirtualMods sends a request to the server to obtain the *vmods* entries for the virtual modifiers specified in the mask, *which,* and waits for a reply.

Virtual modifiers are named by converting their string name to an X Atom and storing the Atom in the *names.vmods* array in an XkbDescRec structure. The position of a name Atom in the *names.vmods* array defines the bit position used to represent the virtual modifier and also the index used when accessing virtual modifier information in arrays: the name in the i-th (0 relative) entry of *names.vmods* is the i-th virtual modifier, represented by the mask (1<<i). Throughout Xkb, various functions have a parameter that is a mask representing virtual modifier choices. In each case, the i-th bit (0 relative) of the mask represents the i-th virtual modifier.

To set the name of a virtual modifier, use *XkbSetNames,* using XkbVirtualModNamesMask in *which* and the name in the *xkb* argument; to retrieve indicator names, use *XkbGetNames.*

For each bit set in *which, XkbGetVirtualMods* updates the corresponding virtual modifier definition in the *server->vmods* array of *xkb.* The *xkb* parameter must be a pointer to a valid Xkb keyboard description. If successful, *XkbGetVirtualMods* returns Success.

Virtual Modifier Names and Masks

Virtual modifiers are named by converting their string name to an X Atom and storing the Atom in the *names.vmods* array in an XkbDescRec structure. The position of a name Atom in the *names.vmods* array defines the bit position used to represent the virtual modifier and also the index used when accessing virtual modifier information in arrays: the name in the i-th (0 relative) entry of *names.vmods*

is the i-th virtual modifier, represented by the mask (1<<i). Throughout Xkb, various functions have a parameter that is a mask representing virtual modifier choices. In each case, the i-th bit (0 relative) of the mask represents the i-th virtual modifier.

To set the name of a virtual modifier, use *XkbSetNames*, using XkbVirtualModNamesMask in *which* and the name in the *xkb* argument; to retrieve indicator names, use *XkbGetNames*.

If the *server* map has not been allocated in the *xkb* parameter, *XkbGetVirtualMods* allocates and initializes it before obtaining the virtual modifier bindings.

If the server does not have a compatible version of Xkb, or the Xkb extension has not been properly initialized, *XkbGetVirtualMods* returns BadMatch. Any errors in allocation cause *XkbGetVirtualMods* to return BadAlloc.

RETURN VALUES

Success The *XkbGetVirtualMods* function returns Success when it successfully updates the corresponding virtual modifier definition in the *server->vmods* array of *xkb*.

STRUCTURES

The complete description of an Xkb keyboard is given by an XkbDescRec. The component structures in the XkbDescRec represent the major Xkb components.

```
typedef struct {
  struct _XDisplay * display;      /* connection to X server */
  unsigned short     flags;        /* private to Xkb, do not modify */
  unsigned short     device_spec;  /* device of interest */
  KeyCode            min_key_code; /* minimum keycode for device */
  KeyCode            max_key_code; /* maximum keycode for device */
  XkbControlsPtr     ctrls;        /* controls */
  XkbServerMapPtr    server;       /* server keymap */
  XkbClientMapPtr    map;          /* client keymap */
  XkbIndicatorPtr    indicators;   /* indicator map */
  XkbNamesPtr        names;        /* names for all components */
  XkbCompatMapPtr    compat;       /* compatibility map */
  XkbGeometryPtr     geom;         /* physical geometry of keyboard */
} XkbDescRec, *XkbDescPtr;
```

The *display* field points to an X display structure. The *flags* field is private to the library: modifying *flags* may yield unpredictable results. The *device_spec* field specifies the device identifier of the keyboard input device, or XkbUseCoreKeyboard, which specifies the core keyboard device. The

min_key_code and *max_key_code* fields specify the least and greatest keycode that can be returned by the keyboard.

Each structure component has a corresponding mask bit that is used in function calls to indicate that the structure should be manipulated in some manner, such as allocating it or freeing it. These masks and their relationships to the fields in the XkbDescRec are shown in Table 1.

Table 1 Mask Bits for XkbDescRec

Mask Bit	XkbDescRec Field	Value
XkbControlsMask	ctrls	(1L<<0)
XkbServerMapMask	server	(1L<<1)
XkbIClientMapMask	map	(1L<<2)
XkbIndicatorMapMask	indicators	(1L<<3)
XkbNamesMask	names	(1L<<4)
XkbCompatMapMask	compat	(1L<<5)
XkbGeometryMask	geom	(1L<<6)
XkbAllComponentsMask	All Fields	(0x7f)

DIAGNOSTICS

BadAlloc Unable to allocate storage

BadMatch A compatible version of Xkb was not available in the server or an argument has correct type and range, but is otherwise invalid

SEE ALSO

XkbGetNames(3), **XkbSetNames**(3)