

NAME

XkbSetCompatMap - Modify the server's compatibility map

SYNOPSIS

```
Bool XkbSetCompatMap (Display *display, unsigned int which, XkbDescPtr xkb, Bool
    update_actions);
```

ARGUMENTS

display

connection to server

which

mask of compat map components to set

xkb source for compat map components

update_actions

True => apply to server's keyboard map

DESCRIPTION

To modify the server's compatibility map, first modify a local copy of the Xkb compatibility map, then call *XkbSetCompatMap*. You may allocate a new compatibility map for this purpose using *XkbAllocCompatMap*. You may also use a compatibility map from another server, although you need to adjust the *device_spec* field in the XkbDescRec accordingly. Note that symbol interpretations in a compatibility map (*sym_interpret*, the vector of XkbSymInterpretRec structures) are also allocated using this same function. *XkbSetCompatMap* copies compatibility map information from the keyboard description in *xkb* to the server specified in *display*'s compatibility map for the device specified by the *device_spec* field of *xkb*. Unless you have specifically modified this field, it is the default keyboard device. *which* specifies the compatibility map components to be set, and is an inclusive OR of the bits shown in Table 1.

Table 1 Compatibility Map Component

Masks

Mask	Value Affecting
XkbSymInterpMask	(1<<0)Symbol interpretations
XkbGroupCompatMask	(1<<1)Group

maps
 XkbAllCompatMask (0x3) All compatibility map
 components

After updating its compatibility map for the specified device, if *update_actions* is True, the server applies the new compatibility map to its entire keyboard for the device to generate a new set of key semantics, compatibility state, and a new core keyboard map. If *update_actions* is False, the new compatibility map is not used to generate any modifications to the current device semantics, state, or core keyboard map. One reason for not applying the compatibility map immediately would be if one server was being configured to match another on a piecemeal basis; the map should not be applied until everything is updated. To force an update at a later time, use *XkbSetCompatMap* specifying *which* as zero and *update_actions* as True. *XkbSetCompatMap* returns True if successful and False if unsuccessful. The server may report problems it encounters when processing the request subsequently via protocol errors.

RETURN VALUES

True The *XkbSetCompatMap* function returns True if successful.

False The *XkbSetCompatMap* function returns False if unsuccessful.

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 2.

Table 2 Mask Bits for
`XkbDescRec`

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

Fields

The `XkbSymInterpretRec` structure specifies a symbol interpretation:

```
typedef struct {
    KeySym    sym;      /* keysym of interest or NULL */
    unsigned char flags; /* XkbSI_AutoRepeat, XkbSI_LockingKey */
    unsigned char match; /* specifies how mods is interpreted */
    unsigned char mods;  /* modifier bits, correspond to eight real modifiers */
    unsigned char virtual_mod; /* 1 modifier to add to key virtual mod map */
    XkbAnyAction act;    /* action to bind to symbol position on key */
} XkbSymInterpretRec; /*XkbSymInterpretPtr;
```

SEE ALSO

XkbSetCompatMap(3)

XKB FUNCTIONS

XkbSetCompatMap(3)

XkbAllocCompatMap(3)