NAME

XkbApplyCompatMapToKey - Apply the new compatibility mapping to an individual key to get its semantics updated

SYNOPSIS

Bool XkbApplyCompatMapToKey (XkbDescPtr *xkb***, KeyCode** *key***, XkbChangesPtr** *changes***);**

ARGUMENTS

xkb keyboard description to be updated

key key to be updated

changes
notes changes to the Xkb keyboard description

DESCRIPTION

XkbApplyCompatMapToKey essentially performs the operation described in Core Keyboard Mapping to Xkb Keyboard Mapping Transformation to a specific key. This updates the behavior, actions, repeat status, and virtual modifier bindings of the key.

Core Keyboard Mapping to Xkb Keyboard Mapping Transformation

When a core protocol keyboard mapping request is received by the server, the server's core keyboard map is updated, and then the Xkb map maintained by the server is updated. Because a client may have explicitly configured some of the Xkb keyboard mapping in the server, this automatic regeneration of the Xkb keyboard mapping from the core protocol keyboard mapping should not modify any components of the Xkb keyboard mapping that were explicitly set by a client. The client must set explicit override controls to prevent this from happening (see Explicit Components-Avoiding Automatic Remapping by the Server). The core-to-Xkb mapping is done as follows:

Explicit Components-Avoiding Automatic Remapping by the Server

Whenever a client remaps the keyboard using core protocol requests, Xkb examines the map to determine likely default values for the components that cannot be specified using the core protocol.

This automatic remapping might replace definitions explicitly requested by an application, so the Xkb keyboard description defines an explicit components mask for each key. Any aspects of the automatic remapping listed in the explicit components mask for a key are not changed by the automatic keyboard mapping.

The explicit components masks are held in the *explicit* field of the server map, which is an array indexed by keycode. Each entry in this array is a mask that is a bitwise inclusive OR of the values shown in Table 1.

Table 1 Explicit Component Masks

Bit in Explicit Mask	Value	Protects Against
ExplicitKeyType1	(1<<0)	Automatic determination of the key type associated with Group1.
ExplicitKeyType2	(1<<1)	Automatic determination of the key type associated with Group2.
ExplicitKeyType3	(1<<2)	Automatic determination of the key type associated with Group3.
ExplicitKeyType4	(1<<3)	Automatic determination of the key type associated with Group4.
ExplicitInterpret	(1<<4)	Application of any of the fields of a symbol interpretation to the key in question.
ExplicitAutoRepeat	(1<<5)	Automatic determination of auto-repeat status for the key, as specified in a symbol interpretation.
ExplicitBehavior	(1<<6)	Automatic assignment of the XkbKB_Lock behavior to the key, if the XkbSI_LockingKey flag is set in a symbol interpretation.
ExplicitVModMap	(1<<7)	Automatic determination of the virtual modifier map for the key based on the actions assigned to the key and the symbol interpretations that match the key.

1. Map the symbols from the keys in the core keyboard map to groups and symbols on keys in the Xkb keyboard map. The core keyboard mapping is of fixed width, so each key in the core mapping has the same number of symbols associated with it. The Xkb mapping allows a different number of symbols to be associated with each key; those symbols may be divided into a different number of groups (1-4) for each key. For each key, this process therefore involves partitioning the fixed number of symbols from the core mapping into a set of variable-length groups with a variable number of symbols in each group. For example, if the core protocol map is of width five, the partition for one key might result in one group with two symbols and another with three

symbols. A different key might result in two groups with two symbols plus a third group with one symbol. The core protocol map requires at least two symbols in each of the first two groups.

1a. For each changed key, determine the number of groups represented in the new core keyboard map. This results in a tentative group count for each key in the Xkb map.

1b. For each changed key, determine the number of symbols in each of the groups found in step 1a. There is one explicit override control associated with each of the four possible groups for each Xkb key, ExplicitKeyType1 through ExplicitKeyType4. If no explicit override control is set for a group, the number of symbols used for that group from the core map is two. If the explicit override control is set for a group on the key, the number of symbols used for that Xkb group from the core map is the width of the Xkb group with one exception: because of the core protocol requirement for at least two symbols in each of groups one and two, the number of symbols used for groups one and two is the maximum of 2 or the width of the Xkb group.

1c. For each changed key, assign the symbols in the core map to the appropriate group on the key. If the total number of symbols required by the Xkb map for a particular key needs more symbols than the core protocol map contains, the additional symbols are taken to be NoSymbol keysyms appended to the end of the core set. If the core map contains more symbols than are needed by the Xkb map, trailing symbols in the core map are discarded. In the absence of an explicit override for group one or two, symbols are assigned in order by group; the first symbols in the core map are assigned to group one, in order, followed by group two, and so on. For example, if the core map contained eight symbols per key, and a particular Xkb map contained 2 symbols for G1 and G2 and three for G3, the symbols would be assigned as (G is group, L is shift level):

G1L1 G1L2 G2L1 G2L2 G3L1 G3L2 G3L3

If an explicit override control is set for group one or two, the symbols are taken from the core set in a somewhat different order. The first four symbols from the core set are assigned to G1L1, G1L2, G2L1, G2L2, respectively. If group one requires more symbols, they are taken next, and then any additional symbols needed by group two. Group three and four symbols are taken in complete sequence after group two. For example, a key with four groups and three symbols in each group would take symbols from the core set in the following order:

G1L1 G1L2 G2L1 G2L2 G1L3 G2L3 G3L1 G3L2 G3L3 G4L1 G4L2 G4L3

As previously noted, the core protocol map requires at lease two symbols in groups one and two. Because of this, if an explicit override control for an Xkb key is set and group one and / or group two is of width one, it is not possible to generate the symbols taken from the core protocol set and assigned to position G1L2 and / or G2L2.

1d. For each group on each changed key, assign a key type appropriate for the symbols in the group.

1e. For each changed key, remove any empty or redundant groups.

At this point, the groups and their associated symbols have been assigned to the corresponding key definitions in the Xkb map.

2. Apply symbol interpretations to modify key operation. This phase is completely skipped if the ExplicitInterpret override control bit is set in the explicit controls mask for the Xkb key (see Explicit Components-Avoiding Automatic Remapping by the Server).

2a. For each symbol on each changed key, attempt to match the symbol and modifiers from the Xkb map to a symbol interpretation describing how to generate the symbol.

2b. When a match is found in step 2a, apply the symbol interpretation to change the semantics associated with the symbol in the Xkb key map. If no match is found, apply a default interpretation.

The symbol interpretations used in step 2 are configurable and may be specified using XkbSymInterpretRec structures referenced by the sym_interpret field of an XkbCompatMapRec.

Symbol Interpretations - the XkbSymInterpretRec Structure

Symbol interpretations are used to guide the X server when it modifies the Xkb keymap in step 2. An initial set of symbol interpretations is loaded by the server when it starts. A client may add new ones using XkbSetCompatMap.

Symbol interpretations result in key semantics being set. When a symbol interpretation is applied, the following components of server key event processing may be modified for the particular key involved:

Virtual modifier map
Auto repeat
Key behavior (may be set to XkbKB_Lock)
Key action

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;
```

If sym is not NULL, it limits the symbol interpretation to keys on which that particular keysym is selected by the modifiers matching the criteria specified by *mods* and *match.* If *sym* is NULL, the interpretation may be applied to any symbol selected on a key when the modifiers match the criteria specified by *mods* and *match.*

match must be one of the values shown in Table 2 and specifies how the real modifiers specified in *mods* are to be interpreted.

Table 2 Symbol Interpretation Match Criteria

Match Criteria	Value	Effect
XkbSI_NoneOf	(0)	None of the bits that are on in mods can be set, but other bits can be.
XkbSI_AnyOfOrNone	(1)	Zero or more of the bits that are on in mods can be set, as well as others.
XkbSI_AnyOf	(2)	One or more of the bits that are on in mods can be set, as well as any others.
XkbSI_AllOf	(3)	All of the bits that are on in mods must be set, but others may be set as well.
XkbSI_Exactly	(4)	All of the bits that are on in mods must be set, and no other bits may be set.

In addition to the above bits, *match* may contain the XkbSI_LevelOneOnly bit, in which case the modifier match criteria specified by *mods* and *match* applies only if *sym* is in level one of its group; otherwise, *mods* and *match* are ignored and the symbol matches a condition where no modifiers are set.

```
#define XkbSI_LevelOneOnly  (0x80)  /* use mods + match only if sym is level 1 */
```

If no matching symbol interpretation is found, the server uses a default interpretation where:

```
sym =          0
flags =        XkbSI_AutoRepeat
match =        XkbSI_AnyOfOrNone
mods =         0
virtual_mod =  XkbNoModifier
act =          SA_NoAction
```

When a matching symbol interpretation is found in step 2a, the interpretation is applied to modify the Xkb map as follows.

The *act* field specifies a single action to be bound to the symbol position; any key event that selects the symbol causes the action to be taken. Valid actions are defined in Key Actions.

If the Xkb keyboard map for the key does not have its ExplicitVModMap control set, the XkbSI_LevelOneOnly bit and symbol position are examined. If the XkbSI_LevelOneOnly bit is not set in *match* or the symbol is in position G1L1, the *virtual_mod* field is examined. If *virtual_mod* is not XkbNoModifier, *virtual_mod* specifies a single virtual modifier to be added to the virtual modifier map for the key. *virtual_mod* is specified as an index in the range [0..15].

If the matching symbol is in position G1L1 of the key, two bits in the flags field potentially specify additional behavior modifications:

```
#define  XkbSI_AutoRepeat  (1<<0)  /* key repeats if sym is in position G1L1 */
#define  XkbSI_LockingKey  (1<<1)  /* set KB_Lock behavior if sym is in psn G1L1 */
```

If the Xkb keyboard map for the key does not have its ExplicitAutoRepeat control set, its auto repeat behavior is set based on the value of the XkbSI_AutoRepeat bit. If the XkbSI_AutoRepeat bit is set, the auto-repeat behavior of the key is turned on; otherwise, it is turned off.

If the Xkb keyboard map for the key does not have its ExplicitBehavior control set, its locking behavior is set based on the value of the XkbSI_LockingKey bit. If XkbSI_LockingKey is set, the key behavior is set to KB_Lock; otherwise, it is turned off.

SEE ALSO

XkbKeyAction(3), **XkbKeyActionEntry**(3), **XkbKeyActionsPtr**(3), **XkbKeyHasActions**(3), **XkbKeyNumActions**(3)