APPENDIX F

Translation Table Syntax

This appendix describes the basic syntax of translation table resources, described in Chapter 12, *Setting Resources*.

This appendix explains some of the more complex aspects of translation table syntax. It probably gives more detail than the average user will need but we've included it to help clarify this rather complicated topic.

F.1 Event Types and Modifiers

The syntax of the translation table is sufficiently general to encompass a wide variety of events and circumstances. Event translations can be specified to handle characteristic user interface idioms like double clicking, dragging, or combining keyboard modifiers with pointer button input. To specify translations that use these features, it is necessary to learn more about the detailed syntax used to specify translations.

An activity susceptible to translation is a sequence of events and modifiers (that perform an action). Events are specified in angle brackets and modifiers precede the event they modify. The legal events that can be specified in a translation are as shown in Table F-1.

Table F-1. Event Types and Their Abbreviations

Event Name	Event Type	Abbreviations/Synonyms
KeyPress	Keyboard	Key , KeyDown
KeyUp	Keyboard	KeyRelease
ButtonPress	Mouse Button	BtnDown
ButtonRelease	Mouse Button	BtnUp

Table F-1. Event Types and Their Abbreviations

Event Name	Event Type	Abbreviations/Synonyms
Btn1Down	Mouse Button Press	
.		
.		
Btn5Down		
Btn1Up	Mouse Button Release	
.		
.		
Btn5Up		
MotionNotify	Mouse Motion	Motion , MouseMoved,PtrMoved
ButtonMotion	Motion w/any Button Down	BtnMotion
Button1Motion	Motion w/Button Down	Btn1Motion
.		.
.		.
Button5Motion		Btn5Motion
EnterNotify	Mouse in Window	Enter, EnterWindow
LeaveNotify		LeaveWindow , Leave
FocusIn	Keyboard Input Focus	
FocusOut		
KeymapNotify	Changed Key Map	Keymap
ColormapNotify	Changed Color Map	Clrmap
Expose	Related Exposure Events	
GraphicsExpose		GrExp
NoExpose		NoExp
VisibilityNotify		Visible
CreateNotify	Window Management	Create

Table F-1. Event Types and Their Abbreviations

Event Name	Event Type	Abbreviations/Synonyms
DestroyNotify		Destroy
UnmapNotify		Unmap
MapNotify		Map
MapRequest		MapReq
ReparentNotify		Reparent
ConfigureNotify		Configure
ConfigureRequest		ConfigureReq
GravityNotify		Grav
ResizeRequest		ResReq
CirculateNotify		Circ
CirculateRequest		CircReq
PropertyNotify		Prop
SelectionClear	Intra-client Selection	SelClr
SelectionRequest		SelReq
SelectionNotify		Select

The possible modifiers of an event are listed in the table. The modifiers Mod1 through Mod5 are highly system-dependent, and may not be implemented by all servers.

Table F-2. Key Modifiers

Event Modifiers	Abbreviation
Ctrl	c
Meta	m
Shift	s
Lock	l
Any	
ANY	

F

Table F-2. Key Modifiers

Event Modifiers	Abbreviation
None	
Mod1	1
.	.
Mod5	5

F.1.1 Detail Field

To provide finer control over the translation process, the event part of the translation can include an additional “detail.” For example, if you want the event to require an additional keystroke, for instance, an A key, or a Control-T, then that keystroke can be specified as a translation detail. The default detail field is *ANY* .

The valid translation details are event-dependent. For example, to specify the above example for keypress events, you would use:

```
<Key>A
```

and:

```
Ctrl<Key>T
```

respectively.

Key fields can be specified by the keysym value, as well as by the keysym symbolic name. For example, the keysym value of the Delete key is 0xffff. Keysym values can be determined by examining the file <*X11/keysymdef.h*> or by using the *xmodmap* client. (See Chapter 14, *Customization Clients*). Unfortunately, with some translations the keysym value may actually be required, since not all keysym symbolic names may be properly interpreted.

F.1.2 Modifiers

Modifiers can be closely controlled to define exactly which events can be specified. For example, if you want the action to be performed by pointer button clicks but not by pointer button clicks with the Control or Shift key down, these limitations can be specified. Similarly, if you don’t care if there are modifiers present, this can also be specified.

Table F-3 lists the available event modifiers.

Table F-3. Event Modifiers and their Meanings

Modifier	Meaning
None *<event>*	No modifiers allowed.
<event>	Doesn’t care. Any modifiers okay.

Table F-3. Event Modifiers and their Meanings

Modifier	Meaning
Mod1 Mod	Mod1 and Mod2, plus any others (i.e., anything that includes m1 and m2).
!Mod1 Mod2<*event*>	Mod1 and Mod2 but nothing else.
Mod1 ~Mod2<*event*>	Mod1 and not Mod2.

F.1.3 Complex Translation Examples

The following translation specifies that function *f* is to be invoked when both the Shift key and the third pointer button are pressed.

```
Shift<Btn3Down>: f()
```

To specify that both the Control and Shift keys are to be pressed use:

```
Ctrl Shift<Btn3Down>: f()
```

To specify an optional repeat count for an activity, put a number in parentheses after the action. The number refers to the whole translation. To make the last example require a double-click, with both Control and Shift keys pressed, use:

```
Ctrl Shift<Btn3Down>(2): f()
```

The server distinguishes between single-clicks and double-clicks based on a pre-programmed timing interval. If a second click occurs before the interval expires, then the event is interpreted as a double-click; otherwise the event is interpreted as two single-clicks. The variable `clickTime` is maintained deep in the internals of X. Unfortunately, thus far there is no way to set this time interval to match user preference. Currently it is set to be 200 milliseconds.

A translation involving two or more clicks can be specified as (2+) in the previous example. In general, a plus sign following the number *n* would mean *n* or more occurrences of the event.

Multiple events can be specified by separating them with commas on the translation line. To indicate pressing button 1, pressing button 2, then releasing button 1, and finally releasing button 2, use:

```
<Btn1Down>,<Btn2Down>,<Btn1Up>,<Btn2Up>: f()
```

Another way to describe this action in English would be to say "while button 1 is down, click button 2." "Meaningless" pointer movement is generally ignored. In the previous case, for example, if pointer motion occurred while the buttons were down, it would not interfere with detection of the event. Thus, inadvertent pointer jiggling will not thwart even the most complex user-input sequences.