X11 INPUT SYNTHESIS EXTENSION PROPOSAL

X Consortium Standard

Larry Woestman, Hewlett Packard

Version 1.0

Abstract

This is a proposal for an extension to the X11 server and Xlib.

Table of Contents

Introduction 3
Conventions Used In This Document 3
Definition Of Terms 3
    Input Actions 3
    User Input Actions 3
What Does This Extension Do? 4
Functions In This Extension 4
    High Level Functions 4
    Low Level Functions 6

Introduction

This is a proposal for an extension to the X11 server and Xlib. It provides two capabilities:

- It allows a client to generate user input actions in the server without requiring a user to be present.
- It also allows a client to control the handling of user input actions by the server.

The capability to allow a client to generate user input actions in the server will be used by some of the X Testing Consortium Xlib tests. Both capabilities will be used by the X Testing Consortium client exerciser program. These capabilities may also be useful in other programs.

This extension requires modification to device-dependent code in the server. Therefore it is not a 'portable' extension as defined by the X11 Server Extensions document. However, the majority of the code and functionality of this extension will be implementation-independent.

Conventions Used In This Document

The naming conventions used in the Xlib documentation are followed with these additions:

- The names of all functions defined in this extension begin with 'XTest', with the first letter of each additional word capitalized.
- The names of the protocol request structures follow the Xlib convention of 'x<name>Req'.
- The names of the protocol request minor type codes follow the Xlib convention of 'X_<name>'.
- The names of all other constants defined in this extension begin with 'XTest', with the rest of the name in upper case letters.
- All constants and structures defined in this extension will have their values specified in the 'xtestext1.h' file (listed in section 5).

Definition Of Terms

Input Actions

Input actions are pointer movements, button presses and releases, and key presses and releases. They can be generated by a user or by a client (using functions in this extension).

User Input Actions

User input actions are input actions that are generated by the user moving a pointing device (typically a mouse), pressing and releasing buttons on the pointing device, and pressing and releasing keys on the keyboard.

What Does This Extension Do?

Without this extension, user input actions are processed by the server, and are converted into normal X events that are sent to the appropriate client or clients.

This extension adds the following capabilities:

- Input actions may be sent from a client to the server to be processed just as if the user had physically performed them. The input actions are provided to the server in the form of X protocol requests defined by this extension. The information provided to the server includes what action should be performed, and how long to delay before processing the action in the server.
- User input actions may be diverted to a client before being processed by the server. The effect on the server is as if the user had performed no input action. The user input actions are provided to the client in the form of X events defined by this extension. The information provided to the client includes what user input action occurred and the delay between this user input action and the previous user input action. The client may then do anything it wishes with this information.
- User input actions may be copied, with one copy going to the server in the normal way, and the other copy being sent to a client as described above.

Functions In This Extension

High Level Functions

These functions are built on top of the low level functions described later.

XTestMovePointer

```
int XTestMovePointer(*display, device_id, delay, x, y, count);
```

display — Specifies the connection to the X server.

device_id — Specifies which pointer device was supposed to have caused the input action. This is a provision for future support of multiple (distinguishable) pointer devices, and should always be set to 0 for now.

delay — Specifies the time (in milliseconds) to wait before each movement of the pointer.

x, y — Specifies the x and y coordinates to move the pointer to relative to the root window for the specified display.

count — Specifies the number of 'delay, x, y' triplets contained in the *delay*, *x* and *y* arrays.

The `XTestMovePointer` function creates input actions to be sent to the the server. The input actions will be accumulated in a request defined by this extension until the request is full or the XTestFlush function is called. They will then be sent to the server. When the input actions are sent to the server, the input actions will cause the server to think that the pointer was moved to the specified position(s), with the specified delay before each input action.

The `XTestMovePointer` function will return -1 if there is an error, and 0 otherwise.

XTestPressButton

`int` **`XTestPressButton`**`(*display, device_id, delay, button_number, button_action);`

display — Specifies the connection to the X server.

device_id — Specifies which button device was supposed to have caused the input action. This is a provision for future support of multiple (distinguishable) button devices, and should always be set to 0 for now.

delay — Specifies the time (in milliseconds) to wait before the input action.

button_number — Specifies which button is being acted upon.

button_action — Specifies the action to be performed (one of *XTestPRESS*, *XTestRELEASE*, or *XTestSTROKE*).

The `XTestPressButton` function creates input actions to be sent to the the server. The input actions will be accumulated in a request defined by this extension until the request is full or the XTestFlush function is called. They will then be sent to the server. When the input actions are sent to the server, the input actions will cause the server to think that the specified button was moved as specified.

The `XTestPressButton` function will return -1 if there is an error, and 0 otherwise.

XTestPressKey

`int` **`XTestPressKey`**`(*display, device_id, delay, keycode, key_action);`

display — Specifies the connection to the X server.

device_id — Specifies which keyboard device was supposed to have caused the input action. This is a provision for future support of multiple (distinguishable) keyboard devices, and should always be set to 0 for now.

delay — Specifies the time (in milliseconds) to wait before the input action.

keycode — Specifies which keycode is being acted upon.

key_action — Specifies the action to be performed (one of *XTestPRESS*, *XTestRELEASE*, or *XTestSTROKE*).

The `XTestPressKey` function creates input actions to be sent to the the server. The input actions will be accumulated in a request defined by this extension until the request is full or the XTestFlush function is called. They will then be sent to the server. When the input actions are sent to the server, the input actions will cause the server to think that the specified key on the keyboard was moved as specified.

The `XTestPressKey` function will return -1 if there is an error, and 0 otherwise.

XTestFlush

`int` **`XTestFlush`**`(*display);`

display Specifies the connection to the X server.

The `XTestFlush` will send any remaining input actions to the server.

The `XTestFlush` function will return -1 if there is an error, and 0 otherwise.

Low Level Functions

XTestGetInput

`int` **`XTestGetInput`**`(`*`display, action_handling`*`);`

display Specifies the connection to the X server.

action_handling Specifies to the server what to do with the user input actions. (one of 0, *XTestPACKED_MOTION* or *XTestPACKED_ACTIONS*; optionally 'or'ed with *XTestEXCLUSIVE*).

The `XTestGetInput` function tells the server to begin putting information about user input actions into events to be sent to the client that called this function. These events can be read via the Xlib `XNextEventf`R function.

The server assigns an event type of *XTestInputActionType* to these events to distinguish them from other events. Since the actual value of the event type may vary depending on how many extensions are included with an X11 implementation, *XTestInputActionType* is a variable that will be contained in the Xlib part of this extension. It may be referenced as follows:

extern int XTestInputActionType;

- An *action_handling* value of 0 causes the server to send one user input action in each *XTestInputActionType* event. This can sometimes cause performance problems.
- An *action_handling* value of *XTestPACKED_ACTIONS* causes the server to pack as many user input actions as possible into a *XTestInputActionType* event. This is needed if user input actions are happening rapidly (such as when the user moves the pointer) to keep performance at a reasonable level.
- An *action_handling* value of *XTestPACKED_MOTION* causes the server to pack only user input actions associated with moving the pointer. This allows the client to receive button and key motions as they happen without waiting for the event to fill up, while still keeping performance at a reasonable level.
- An *action_handling* value with *XTestEXCLUSIVE* 'or'ed in causes the server to send user input actions only to the client. The effect on the server is as if the user had performed no input actions.
- An *action_handling* value without *XTestEXCLUSIVE* causes the server to copy user input actions, sending one copy to the client, and handling the other copy normally (as it would if this extension were not installed).

There are four types of input actions that are passed from the server to the client. They are:

key/button~state~change This type of input action contains the keycode of the key or button that changed state; whether the key or

button is up or down, and the time delay between this input action and the previous input action.

pointer~motions — This type of input action contains information about the motion of the pointer when the pointer has only moved a short distance. If the pointer has moved a long distance, the pointer jump input action is used.

pointer~jumps — This type of input action contains information about the motion of the pointer when the pointer has moved a long distance.

delays — This type of input action is used when the delay between input actions is too large to be held in the other input actions.

The `XTestGetInput` function will return -1 if there is an error, and 0 otherwise.

An error code of *BadAccess* means that another client has already requested that user input actions be sent to it.

XTestStopInput

```
int XTestStopInput(*display);
```

display Specifies the connection to the X server.

The `XTestStopInput` function tells the server to stop putting information about user input actions into events. The server will process user input actions normally (as it would if this extension were not in the server).

The `XTestStopInput` function will return -1 if there is an error, and 0 otherwise.

An error code of *BadAccess* means that a request was made to stop input when input has never been started.

XTestFakeInput

```
int XTestFakeInput(*display, *action_list_addr, action_list_size,
ack_flag);
```

display — Specifies the connection to the X server.

action_list_addr — Specifies the address of an list of input actions to be sent to the server.

action_list_size — Specifies the size (in bytes) of the list of input actions. It may be no larger than *XTestMAX_ACTION_LIST_SIZE* bytes.

ack_flag — Specifies whether the server needs to send an event to indicate that its input action buffer is empty (one of *XTestFAKE_ACK_NOT_NEEDED* or *XTestFAKE_ACK_REQUEST*).

The `XTestFakeInput` function tells the server to take the specified user input actions and process them as if the user had physically performed them.

The server can only accept a limited number of input actions at one time. This limit can be determined by the `XTestQueryInputSize` function in this extension.

The client should set *ack_flag* to *XTestFAKE_ACK_NOT_NEEDED* on calls to *XTestFakeInput* that do not reach this limit.

The client should set *ack_flag* to *XTestFAKE_ACK_REQUEST* on the call to *XTestFakeInput* that reaches this limit.

When the server sees an *ack_flag* value of *XTestFAKE_ACK_REQUEST* it finishes processing its input action buffer, then sends an event with type *XTestFakeAckType* to the client. When the client reads this event, it knows that it is safe to resume sending input actions to the server.

Since the actual value of the event type may vary depending on how many extensions are included with an X11 implementation, *XTestFakeAckType* is a variable that is contained in the Xlib part of this extension. It may be referenced as follows:

extern int XTestFakeAckType;

There are four types of input actions that are passed from the client to the server. They are:

key/button~state~change	This type of input action contains the keycode of the key or button that is to change state; whether the key or button is to be up or down, and the time to delay before changing the state of the key or button.
pointer~motions	This type of input action contains information about the motion of the pointer when the pointer is to be moved a short distance, and the time to delay before moving the pointer. If the pointer is to be moved a long distance, the pointer jump input action must be used.
pointer~jumps	This type of input action contains information about the motion of the pointer when the pointer is to be moved a long distance, and the time to delay before moving the pointer.
delays	This type of input action is used when the delay between input actions is too large to be held in the other input actions.

The `XTestFakeInput` function will return -1 if there is an error, and 0 otherwise.

An error code of \fIBadAccess\fR means that another client has already sent user input actions to the server, and the server has not finished processing the user input actions.

XTestQueryInputSize

`int` **`XTestQueryInputSize`**`(`*`*display, size_return`*`);`

display	Specifies the connection to the X server.
size_return	Returns the number of input actions that the server's input action buffer can hold.

The `XTestQueryInputSize` function asks the server to return the number of input actions that it can hold in its input action buffer in the unsigned long pointed to by \fIsize_return\fR.

The `XTestQueryInputSize` function will return -1 if there is an error, and 0 otherwise.

XTestReset

```
int XTestReset(*display);
```

display Specifies the connection to the X server.

The `XTestReset` function tells the server to set everything having to do with this extension back to its initial state. After this call the server will act as if this extension were not installed until one of the extension functions is called by a client. This function is not normally needed, but is included in case a client wishes to clean up the server state, such as after a serious error.

The `XTestReset` function will return -1 if there is an error, and 0 otherwise.