XTEST Extension Protocol

X Consortium Standard

Kieron Drake, UniSoft Ltd.

XTEST Extension Protocol: X Consortium Standard

by Kieron Drake

X Version 11, Release 7.7

Version 2.2

Copyright © 1992 UniSoft Group Ltd.
Copyright © 1992, 1994 X Consortium

Permission to use, copy, modify, and distribute this documentation for any purpose and without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies. UniSoft makes no representations about the suitability for any purpose of the information in this document. This documentation is provided "as is" without express or implied warranty.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE X CONSORTIUM BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Except as contained in this notice, the name of the X Consortium shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization from the X Consortium.

Table of Contents

Chapter 1. Overview

This extension is a minimal set of client and server extensions required to completely test the X11 server with no user intervention.

This extension is not intended to support general journaling and playback of user actions. This is a difficult area [XTrap, 89] as it attempts to synchronize synthetic user interactions with their effects; it is at the higher level of dialogue recording/playback rather than at the strictly lexical level. We are interested only in the latter, simpler, case. A more detailed discussion and justification of the extension functionality is given in [Drake, 91].

We are aiming only to provide a minimum set of facilities that solve immediate testing and validation problems. The testing extension itself needs testing, where possible, and so should be as simple as possible.

We have also tried to:

- Confine the extension to an appropriate high level within the server to minimize portability problems. In practice this means that the extension should be at the DIX level or use the DIX/DDX interface, or both. This has effects, in particular, on the level at which "input synthesis" can occur.
- Minimize the changes required in the rest of the server.
- Minimize performance penalties on normal server operation.

Chapter 2. Description

The functions provided by this extension fall into two groups:

Client Operations	These routines manipulate otherwise hidden client-side behavior. The actual implementation will depend on the details of the actual language binding and what degree of request buffering, GContext caching, and so on, is provided. In the C binding, defined in "XTEST Extension Library", routines are provided to access the internals of two opaque data structures -- `GC`s and `Visual`s -- and to discard any requests pending within the output buffer of a connection. The exact details can be expected to differ for other language bindings.
Server Requests	The first of these requests is similar to that provided in most extensions: it allows a client to specify a major and minor version number to the server and for the server to respond with major and minor versions of its own. The remaining two requests allow the following: • Access to an otherwise "write-only" server resource: the cursor associated with a given window • Perhaps most importantly, limited synthesis of input device events, almost as if a cooperative user had moved the pointing device or pressed a key or button.

Chapter 3. Types

The following types are used in the request and event definitions in subsequent sections:

FAKE_EVENT_TYPE { `KeyPress`, `KeyRelease`, `MotionNotify`, `ButtonPress`, `ButtonRelease` }

FAKE_EVENT [type: FAKE_EVENT_TYPE,
detail: BYTE,
time: TIME,
root: WINDOW,
rootX, rootY: INT16]

CURSOR { `CurrentCursor, None` } or a cursor as defined by the X11 Protocol.

Chapter 4. Client Operations

These are abstract definitions of functionality. They refer to client-side objects such as "GC" and "VISUAL" that are quoted to denote their abstract nature. Concrete versions of these functions are defined only for particular language bindings. In some circumstances a particular language binding may not implement the relevant abstract type or may provide it as a transparent, rather than opaque, type, with the result that the corresponding function does not make sense or is not required, respectively.

`XTestSetGContextOfGC`

gc: "GC"
gid: GCONTEXT

Sets the GCONTEXT within the "GC" gc to have the value specified by gid.

`XTestSetVisualIDOfVisual`

visual: "VISUAL"
visualid: VISUALID

Sets the VISUALID within the "VISUAL" visual to have the value specified by visualid.

`XTestDiscard`

dpy: "CONNECTION"
=>
status: BOOL

Discards any requests that are present in the request buffer associated with the "CONNECTION" dpy. The status returned is `True` if there were one or more requests in the buffer and `False` otherwise.

Chapter 5. Server Requests

`XTestGetVersion`

clientMajorVersion: CARD16
clientMinorVersion: CARD16
=>
serverMajorVersion: CARD16
serverMinorVersion: CARD16
Errors: `Length`

This request can be used to ensure that the server version of the XTEST extension is usable by the client. This document defines major version two (2), minor version one (1).

`XTestCompareCursor`

window: WINDOW
cursor-id: CURSOR or `CurrentCursor` or `None`
=>
same: BOOL
Errors: `Window`, `Length`, `Cursor`

This request looks up the cursor associated with the window and compares it with either the null cursor if cursor-id is `None` , or the current cursor (that is, the one being displayed), or the cursor whose ID is cursor-id, and returns the result of the comparison in same.

`XTestFakeInput`

events: LISTofFAKE_EVENT
Errors: `Window`, `Length`, `Alloc`, `Value`

This request simulates the limited set of core protocol events within the set FAKE_EVENT_TYPE. Only the following event fields, defined in FAKE_EVENT, are interpreted:

type	This must be one of `KeyPress`, `KeyRelease`, `MotionNotify`, `ButtonPress`, or `ButtonRelease`, or else a `Value` error occurs.
detail	For key events, this field is interpreted as the physical keycode. If the keycode is less than min-keycode or greater than max-keycode, as returned in the connection setup, then a `Value` error occurs. For button events, this field is interpreted as the physical (or core) button, meaning it will be mapped to the corresponding logical button according to the most recent `SetPointerMapping` request. If the button number is less than one or greater than the number of physi-

	cal buttons, then a `Value` error occurs. For motion events, if this field is `True` , then rootX and rootY are relative distances from the current pointer location; if this field is `False`, then they are absolute positions.
time	This is either `CurrentTime` (meaning no delay) or the delay in milliseconds that the server should wait before simulating this event. No other requests from this client will be processed until this delay, if any, has expired and subsequent processing of the simulated event has been completed.
root	In the case of motion events this field is the ID of the root window on which the new motion is to take place. If `None` is specified, the root window of the screen the pointer is currently on is used instead. If this field is not a valid window, then a `Window` error occurs.
rootX & *rootY*	In the case of motion events these fields indicate relative distance or absolute pointer coordinates, according to the setting of detail. If the specified coordinates are off-screen, the closest on-screen coordinates will be substituted.

When the simulated event(s) are processed, they cause event propagation, passive grab activation, and so on, just as if the corresponding input device action had occurred. However, motion events might not be recorded in the motion history buffer.

For the currently supported event types, the event list must have length one, otherwise a `BadLength` error occurs.

`XTestGrabControl`

impervious: BOOL

If impervious is `True`, then the executing client becomes impervious to server grabs; that is, it can continue executing requests even if another client grabs the server. If impervious is `False`, then the executing client returns to the normal state of being susceptible to server grabs.

Chapter 6. Encoding

Please refer to the X11 Protocol Encoding document as this document uses conventions established there.

The name of this extension is "XTEST".

New Types

```
FAKE_EVENT_TYPE
     2     KeyPress
     3     KeyRelease
     4     ButtonPress
     5     ButtonRelease
     6     MotionNotify
```

NOTE that the above values are defined to be the same as those for the corresponding core protocol event types.

Requests

```
XTestGetVersion
     1     CARD8                opcode
     1     0                    xtest opcode
     2     2                    request length
     1     CARD8                client major version
     1                          unused
     2     CARD16               client minor version
=>
     1     1                    Reply
     1     CARD8                server major version
     2     CARD16               sequence number
     4     0                    reply length
     2     CARD16               server minor version
     22                         unused
```

```
XTestCompareCursor
     1     CARD8                opcode
     1     1                    xtest opcode
     2     3                    request length
     4     WINDOW               window
     4     CURSOR               cursor-id
           0     None
           1     CurrentCursor
=>
     1     1                    Reply
     1     BOOL                 cursors are the same
     2     CARD16               sequence number
     4     0                    reply length
```

```
     24                         unused
```

```
XTestFakeInput
     1     CARD8                opcode
     1     2                    xtest opcode
     2     1+(1*8)              request length
     1     FAKE_EVENT_TYPE      fake device event type
     1     BYTE                 detail: button or keycode
     2                          unused
     4     TIME                 delay (milliseconds)
           0     CurrentTime
     4     WINDOW               root window for MotionNotify
           0     None
     8                          unused
     2     INT16                x position for MotionNotify
     2     INT16                y position for MotionNotify
     8                          unused
```

```
XTestGrabControl
     1     CARD8                opcode
     1     3                    xtest opcode
     2     2                    request length
     1     BOOL                 impervious
     3                          unused
```

Chapter 7. References

Annicchiarico, D., et al., *XTrap: The XTrap Architecture*. Digital Equipment Corporation, July 1991.

Drake, K. J., *Some Proposals for a Minimum X11 Testing Extension*. UniSoft Ltd., June 1991.