XTEST Extension Library

X Consortium Standard

Kieron Drake, UniSoft Ltd.

XTEST Extension Library: X Consortium Standard

by Kieron Drake

X Version 11, Release 7.7

Version 2.2

Copyright © 1992 UniSoft Group Ltd.

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.

Copyright © 1992,1994 X Consortium

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.

X Window System is a trademark of The Open Group.

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 section 7, routines are provided to access the internals of two opaque data structures - GCs 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. C Language Binding

The C functions either provide direct access to the protocol and add no additional semantics to those defined in section 5 or they correspond directly to the abstract descriptions of client operations in section 4.

All XTEST extension functions and procedures, and all manifest constants and macros, will start with the string "XTest". All operations are classified as server/client (Server) or client-only (Client). All routines that have return type Status will return nonzero for "success" and zero for "failure." Even if the XTEST extension is supported, the server may withdraw such facilities arbitrarily; in which case they will subsequently return zero.

The include file for this extension is <X11/extensions/XTest.h>.

`Bool` **`XTestQueryExtension`**`(` *`*display`*`,` *`*event_base`*`,` *`*error_base`*`,` *`*major_version`*`,` *`*minor_version`*`);`

`XTestQueryExtension` returns `True` if the specified display supports the XTEST extension, else `False` If the extension is supported, *event_base would be set to the event number for the first event for this extension and *error_base would be set to the error number for the first error for this extension. As no errors or events are defined for this version of the extension, the values returned here are not defined (nor useful). If the extension is supported, *major_version and *minor_version are set to the major and minor version numbers of the extension supported by the display. Otherwise, none of the arguments are set.

`Bool` **`XTestCompareCursorWithWindow`**`(` *`*display`*`,` *`*window`*`,` *`cursor`*`);`

If the extension is supported, `XTestCompareCursorWithWindow` performs a comparison of the cursor whose ID is specified by cursor (which may be `None` with the cursor of the window specified by window returning `True` if they are the same and `False` otherwise. If the extension is not supported, then the request is ignored and zero is returned.

`Bool`**`XTestCompareCurrentCursorWithWindow`**`(` *`*display`*`,` *`window`*`);`

If the extension is supported, `XTestCompareCurrentCursorWithWindow` performs a comparison of the current cursor with the cursor of the specified window returning `True` if they are the same and `False` otherwise. If the extension is not supported, then the request is ignored and zero is returned.

`XTestFakeKeyEvent``(` *`*display`*`,` *`keycode`*`,` *`is_press`*`,` *`delay`*`);`

If the extension is supported, `XTestFakeKeyEvent` requests the server to simulate either a `KeyPress` (if is_press is `True` or a `KeyRelease` (if is_press is `False` of the key with the specified keycode; otherwise, the request is ignored.

If the extension is supported, the simulated event will not be processed until delay milliseconds after the request is received (if delay is `CurrentTime` then this is interpreted as no delay at all). 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.

`XTestFakeButtonEvent``(` *`*display`*`,` *`button`*`,` *`is_press`*`,` *`delay`*`);`

If the extension is supported, `XTestFakeButtonEvent` requests the server to simulate either a `ButtonPress` (if is_press is `True` or a `ButtonRelease` (if is_press is `False` of the logical button numbered by the specified button; otherwise, the request is ignored.

If the extension is supported, the simulated event will not be processed until delay milliseconds after the request is received (if delay is `CurrentTime` then this is interpreted as no delay at all). 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.

XTestFakeMotionEvent(**display*, *screen_number*, *x*, *y*, *delay*);

If the extension is supported, `XTestFakeMotionEvent` requests the server to simulate a movement of the pointer to the specified position (x, y) on the root window of screen_number; otherwise, the request is ignored. If screen_number is -1, the current screen (that the pointer is on) is used.

If the extension is supported, the simulated event will not be processed until delay milliseconds after the request is received (if delay is `CurrentTime` then this is interpreted as no delay at all). 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.

XTestFakeRelativeMotionEvent(**display*, *screen_number*, *x*, *y*, *delay*);

If the extension is supported, `XTestFakeRelativeMotionEvent` requests the server to simulate a movement of the pointer by the specified offsets (x, y) relative to the current pointer position on screen_number; otherwise, the request is ignored. If screen_number is -1, the current screen (that the pointer is on) is used.

If the extension is supported, the simulated event will not be processed until delay milliseconds after the request is received (if delay is `CurrentTime` then this is interpreted as no delay at all). 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.

XTestGrabControl(**display*, *impervious*);

If impervious is `True` then the executing client becomes impervious to server grabs. If impervious is `False` then the executing client returns to the normal state of being susceptible to server grabs.

Bool **XTestSetGContextOfGC**(*gc*, *gid*);

`XTestSetGContextOfGC` sets the GContext within the opaque datatype referenced by gc to be that specified by gid.

XTestSetVisualIDOfVisual(**visual*, *visualid*);

`XTestSetVisualIDOfVisual` sets the VisualID within the opaque datatype referenced by visual to be that specified by visualid.

Bool **XTestDiscard**(**display*);

`XTestDiscard` discards any requests within the output buffer for the specified display. It returns `True` if any requests were discarded; otherwise, it returns `False`

Chapter 4. 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.