Extending X for Recording

Version 1.3

DRAFT - not yet an X Consortium standard

Martha Zimet
Network Computing Devices, Inc.

Copyright © 1994 Network Computing Devices, Inc.

Permission to use, copy, modify, distribute, and sell this documentation for any purpose is hereby granted without fee, provided that the above copyright notice and this permission notice appear in all copies. Network Computing Devices, Inc. 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.

1. Introduction

Several proposals have been written over the past few years that address some of the issues surrounding the recording and playback of user actions in the X Window System[1]:

- *Some Proposals for a Minimal X11 testing Extension*, Kieron Drake, UniSoft Ltd., April 1991
- *X11 Input Synthesis Extension Proposal*, Larry Woestman, Hewlett Packard, November 1991
- *XTrap Architecture*, Dick Annicchiario, et al, Digital Equipment Corporation, July 1991
- *XTest Extension Recording Specification*, Yochanan Slonim, Mercury Interactive, December 1992

This document both unifies and extends the previous diverse approaches to generate a proposal for an X extension that provides support for the recording of all core X protocol and arbitrary extension protocol.

Input synthesis, or playback, has already been implemented in the XTest extension, an X Consortium standard, therefore, this extension is limited to recording. This proposal addresses the issue of synchronization, since a recording solution would be incomplete without it. The requirement to provide synchronization of contextual information comes from the playback mechanism. That is, there needs to be a way to interleave device or user events with their consequences. Hence, contextual information must be captured during recording.

Since it is impossible to predict in advance what synchronization information will be required by a particular application, the proposed extension takes a "*broad brush*" approach, whereby facilities will exist to both record and filter core X protocol and arbitrary extension protocol. As such, the proposal does not enforce a specific synchronization methodology (eg. window mapping/unmappping, cursor changes, text strings, etc.).

The name of the extension, as seen by ListExtensions and QueryExtension, is "RECORD."

Several people have provided input to this document:

Robert Chesler	Absol-puter
Kieron Drake	UniSoft Ltd.
Marc Evans	Synergytics
Jim Fulton	NCD
Ken Miller	DEC
Bob Scheifler	X Consortium

1. *X Window System* is a trademark of X Consortium, Inc.

1.1. Goals

- To implement an X Consortium standard for recording and synchronization, whereby both device events and contextual synchronization information in the form of device event consequences are recorded.
- To record contextual information used in synchronized playback without beforehand knowledge of the application which is being recorded.
- To provide the ability to work with an arbitrary set of X extension protocol.

1.2. Requirements

The extension should:

1. not be dependent on other clients or extensions for its operation
2. not significantly impact performance
3. should support the recording of all device input (core devices and XInput devices)
4. should be extensible
5. should support the recording of contextual information for user events
6. follow an event-driven synchronization model
7. handle byte-order and provide word-size independence
8. support multiple client connections

2. Design

2.1. Overview

The mechanism used by this extension for recording is to intercept core X protocol and arbitrary X extension protocol entirely within the X server itself. When the extension has been requested by one or more clients to intercept specific protocol, the protocol data is formatted and returned to the recording clients. The extension provides a mechanism for capturing all events, including input device events that go to no clients, that is analogous to a client expressing ''interest'' in all events, in all windows, including the root window. Event filtering in the extension provides a mechanism for feeding device events to recording clients; it does not provide a mechanism for in-place, synchronous event substitution, modification, or withholding. It is assumed that this functionality will be provided in another extension or in a later version of this extension.

A **Configuration** resource is added by the extension to the set of resources managed by the server. Although any program with multiple paths open to the server is viewed as multiple clients by the X Window System protocol[1], the recommended communication model for a recording client is to open two connections to the server and use one for configuration control and the other for reading protocol data. The ''control'' connection can execute requests to obtain information about the supported protocol version, create and destroy configurations, specify protocol types to be intercepted and clients to be recorded, query the current state of a configuration, and to stop interception and reporting of protocol data. The ''data'' connection can execute a request to start interception and reporting of specified protocol for a particular configuration. Intercepted protocol is sent back on the data connection, and may be sent by more than one reply packet. Individual requests, replies, events or errors intercepted by the extension on behalf of recording clients may not be split across reply packets.

The extension manages the intercepted protocol data through a record format that describes and extends the intercepted protocol data. The data can be unpackaged by any client that follows the data formatting protocol of the extension. This data format wraps the core X protocol. Aligned extension-specific information is at the front of the data.

---

1. Scheifler, Robert W. 'X Window System Protocol Version 11'

2.2. Types

The following new types are used in the request definitions in the following section.

CONFIGURATION: XID

The CONFIGURATION type defines a client-side handle for a server **Configuration**.

RANGE: [*first*, *last*: CARD8]
RANGE16: [*first*, *last*: CARD16]
EXTRANGE: [*major*: CARD8, *minor*: LISTofRANGE16]

RECORDFLAGS: [*core-requests*: LISTofRANGE
core-replies: LISTofRANGE
ext-requests: LISTofEXTRANGE
ext-replies: LISTofEXTRANGE
events: LISTofRANGE
unsent-events: LISTofRANGE
errors: LISTofRANGE]

The **RECORDFLAGS** structure contains the protocol values to be intercepted. Typically, this structure is sent by recording clients over the control connection when creating or modifying a configuration.

core-requests
Specifies core X protocol requests.

core-replies
Specifies core X protocol replies.

ext-requests
Specifies arbitrary X extension requests.

ext-replies
Specifies arbitrary X extension replies.

events
Specifies core X protocol events and arbitrary extension events.

unsent-events
Specifies core device events not delivered to any clients.

errors
Specifies X protocol errors and arbitrary extension errors.

RECORDSTATE:	[*enabled*:	BOOL
	intercepted-clients:	LISTofCARD32
	intercepted-protocol:	LISTofRECORDFLAGS]

This structure specifies the current state of a configuration environment. **RECORDSTATE** is returned when the configuration state has been queried, typically by a recording client over the control connection.

The enabled field specifies the state of data transfer between the extension and the recording client, and is either enabled (1) or disabled (0). The default state is disabled. When enabled, all core X protocol and extension protocol received from (xRequest) or sent to (xReply, xError, xEvent) a particular client, and requested by the recording client to be intercepted, will be reported to the recording client over the data connection.

The intercepted-clients field defines the list of clients currently being recorded, where each recorded client is identified by its resource base. The intercepted-protocol field specifies the protocol to be intercepted for each recorded client.

2.3. Errors

Configuration:
This error is returned if the value for a Configuration argument in a request does not name a valid CONFIGURATION.

3. Protocol

3.1. Requests

QueryVersion

major-version minor-version: CARD16

→

major-version minor-version: CARD16

Errors:

This request specifies the RECORD extension protocol version the client would like to use. When the specified protocol version is supported by the extension, the protocol version the server expects from the client is returned. Clients should use this request before other RECORD extension requests. If this request is not the first extension request executed, the protocol version used by the extension is implementation dependent.

This request determines whether or not the RECORD extension protocol version specified by the client is supported by the extension. If the extension supports the version specified by the client, this version number should be returned. If the client has requested a higher version than is supported by the server, the server's highest version should be returned. Otherwise, if the client has requested a lower version than is supported by the server, the server's lowest version should be returned. This document defines major version one (1), minor version zero (0).

CreateConfiguration

id: CONFIGURATION

id-base: CARD32

flags: RECORDFLAGS

Errors: **Match**, **Value**, **Alloc**

This request creates a new **Configuration** object within the server and assigns identifier id to it. Typically, this request is used by a recording client over the control connection to express interest in specific core X protocol and X extension protocol to be intercepted by the extension for a particular client.

The id-base argument is the resource identifier base sent to a client from the server in the connection setup reply, and hence, identifies the client to be recorded. An id-base of zero (0) specifies the configuration values will apply to the set of clients that includes all current client connections and all future client connections. The flags argument specifies the values of the configuration environment, respective to the recorded client or clients.

The error **Match** is returned when the id-base argument does not identify a client connection. **Value** is returned when a value specified falls outside of the accepted range. **Alloc** is returned when the extension is unable to allocate the necessary resources.

ChangeConfiguration

configuration: CONFIGURATION

id-base: CARD32

flags: RECORDFLAGS

add: BOOL

Errors: **Match**, **Value**, **Alloc**, **Configuration**

Typically, this request is used by a recording client over the control connection to modify the specific protocol and to add or delete clients from the set of clients to be intercepted by the extension.

The id-base argument is the resource identifier base sent to a client from the server in the connection setup reply, and hence, identifies the client to be recorded. An id-base of zero (0) specifies the configuration

values will apply to the set of clients that includes all current client connections and all future client connections. The flags argument specifies the configuration values, respective to the recorded client or clients.

The add argument specifies whether to add or delete the client, specified by id-base, from the configuration. When add is ‘‘True’ and the client is not associated with the configuration, the configuration values are assigned to the client and the client is associated with the configuration. When add is ‘‘True’ and the client is already associated with the configuration, the new configuration values specified by the flags argument are assigned to the client. When add is ‘‘False’ the client is deleted from the set of clients associated with the configuration.

The error **Match** is returned when the id-base argument does not identify a client connection. **Value** is returned when a value specified falls outside of the accepted range. **Alloc** is returned when the extension is unable to allocate the necessary resources. When the configuration argument is not valid, a **Configuration** error is returned.

GetConfiguration

configuration: CONFIGURATION

→+

num-clients: CARD32

state: RECORDSTATE

Errors: **Configuration**

This request queries the current state of a configuration and is typically used by a recording client over the control connection. This request always returns the number of clients currently being recorded. When the number of recorded clients is greater than zero (0), the current configuration environment is returned, consisting of the data transfer state, the list of recorded clients and the protocol values intercepted for each client. When the number of recorded clients is equal to zero (0), the current environment is returned, consisting only of the data transfer state.

When the configuration argument is not valid, a **Configuration** error is returned.

EnableConfiguration

configuration: CONFIGURATION

→+

replies-following-hint: CARD32

direction: {FromClient, FromServer}

client-swapped: BOOL

id-base: CARD32

client-seq: CARD32

client-req: CARD32

data: LISTofCARD8

Errors: **Match**, **Alloc**, **Configuration**

This request specifies to enable data transfer between the recording client and the extension, and returns the protocol data the recording client has previously expressed interest in. Typically, this request will be executed by the recording client over the data connection.

Once the extension completes processing this request, it begins intercepting and reporting to the recording client all core and extension protocol received from or sent to clients associated with the configuration that the recording client has expressed interest in. All intercepted protocol data is returned in the byte-order of the recorded client. Therefore, recording clients are responsible for all byte swapping, if required. More than one recording client cannot enable data transfer on the same configuration at the same time.

The replies-following-hint contains a positive value that specifies the number of replies that are likely, but not required, to follow. When the disable request is made over the control connection, the reply is sent over the data connection with replies-following-hint set to zero, indicating the end of the reply sequence. Direction is set to the direction of the protocol data, which is either from the client (xRequest), or from the server (xReply, xError, xEvent). Client-swapped is ''True'' if the byte order of the client being recorded is swapped relative to the server; otherwise, client-swapped is ''False.'' Id-base is the resource identifier base sent to the client from the server in the connection setup reply, and hence, identifies the client being recorded. Client-seq is set to the sequence number of the request just processed by the server on behalf of the recorded client and client-req is set to the number of the request.

The error **Match** is returned when data transfer is already enabled. **Alloc** is returned when the extension is unable to allocate the necessary resources. When the configuration argument is not valid, a **Configuration** error is returned.

DisableConfiguration

configuration: CONFIGURATION

Errors: **Value**, **Alloc**, **Configuration**

This request is typically executed by the recording client over the control connection. This request directs the extension to discontinue data transfer between the extension and the recording client and disable protocol reporting on the data connection that is currently enabled. Once the extension completes processing this request, no protocol received from or sent to the client will be reported.

The error **Alloc** is returned when the extension is unable to allocate the necessary resources. When the configuration argument is not valid, a **Configuration** error is returned.

FreeConfiguration

configuration: CONFIGURATION

Errors: **Configuration**

This request deletes the association between the resource ID and the configuration and destroys the configuration. If enabled for the configuration, data transfer between the extension and any recording clients will be discontinued. All client connections that have enabled data transfer for the configuration will be released.

A configuration is destroyed automatically when the connection to the creating client is closed down and the close-down mode is **DestroyAll**. When the configuration argument is not valid, a **Configuration** error is returned.