Multi-Buffering Application Program Interface, Version 3.2

B

This appendix describes the C-language application program interface (API) to the Multi–Buffering (MBX) extension.[1] These routines provide direct access to the protocol and add no semantics.

This appendix assumes that you are familiar with the MBX protocol described in the MIT standard, *Extending X for Double-Buffering and Multi-Buffering, and Stereo, Version 3.2*. See "X Consortium Extensions" on page 4 for information on how to access this standard.

Throughout this appendix, the file path names given are relative to `/usr/openwin`.

Library File

These API routines can be accessed by dynamically linking with the shared object file, `lib/libXext.so`.

Note – Although a statically linkable version of this same library, `libXext.a`, is available in the same directory, static linking is not recommended because this reduces application compatibility with future releases.

---

1. This document is derived from the document *Multi-Buffering Application Program Interface* by David P. Wiggins (dwig@sr71.b11.ingr.com), Intergraph Corporation, Version 1.0.

B

Header File

The header file for this extension is `include/X11/extensions/multibuf.h`. This file defines the following types, constants, structures, and functions.

New Routines

The following routines are added by this API:

- `XmbufQueryExtension`
- `XmbufGetVersion`
- `XmbufCreateBuffers`
- `XmbufDestroyBuffers`
- `XmbufDisplayBuffers`
- `XmbufGetWindowAttributes`
- `XmbufChangeWindowAttributes`
- `XmbufGetBufferAttributes`
- `XmbufChangeBufferAttributes`
- `XmbufGetScreenInfo`

Note – `XmbufCreateStereoWindow` is not supported in SunSoft's MBX implementation.

New Types

Buffer identifiers are held in a new drawable type, `Multibuffer`. A `Multibuffer` can be substituted in all X calls where a `Drawable` is specified.

New Constants

The following constants are defined in the `multibuf.h` header file.

Event Type Constants

- `MultibufferClobberNotify`
- `MultibufferUpdateNotify`

Error Constants

- `MultibufferBadBuffer`

Update Action Constants

- `MultibufferUpdateActionUndefined`
- `MultibufferUpdateActionBackground`
- `MultibufferUpdateActionUntouched`
- `MultibufferUpdateActionCopied`

Update Hint Constants

- `MultibufferUpdateHintFrequent`
- `MultibufferUpdateHintIntermittent`
- `MultibufferUpdateHintStatic`

Window Mode Constants

- `MultibufferModeMono`

Note – SunSoft's MBX implementation does not support the window mode constant `MultibufferModeStereo`, and the window side constants `MultibufferSideMono`, `MultibufferSideLeft`, and `MultibufferSideRight`

Event Mask Constants

- `MultibufferClobberNotifyMask`
- `MultibufferUpdateNotifyMask`

Valuemask Constants

- `MultibufferWindowUpdateHint`
- `MultibufferBufferEventMask`

B

Clobber State Constants

- `MultibufferUnclobbered`
- `MultibufferPartiallyClobbered`
- `MultibufferFullyClobbered`

New Structures

Several new structure types are defined. Most are introduced in the function discussion of the function that requires a structure as a parameter. The following structures are not parameters in any of the functions discussed in "MBX Functions" on page 111.

MultibufferClobberNotify Event

```
typedef struct {
    int type;       /* = mbuf_event_base + MultibufferClobberNotify */
    unsigned long serial; /* # of last request processed by server */
    int send_event;    /* true if this came from a SendEvent request */
    Display *display;      /* Display the event was read from */
    Multibuffer buffer;    /* buffer of event */
    int state;             /* see Clobber state constants above */
} XmbufClobberNotifyEvent;
```

MultibufferUpdateNotify Event

```
typedef struct {
    int type;        /* = mbuf_event_base + MultibufferUpdateNotify */
    unsigned long serial; /* # of last request processed by server */
    int send_event; /* true if this came from a SendEvent request */
    Display *display;      /* Display the event was read from */
    Multibuffer buffer;    /* buffer of event */
} XmbufUpdateNotifyEvent;
```

MBX Functions

The following functions generate MBX protocol requests. Except for `XmbufQueryExtension`, if any of them are called with a display that does not support the MBX extension, the `ExtensionErrorHandler` (registered by `XSetExtensionErrorHandler`) is called. If the `ExtensionErrorHandler` returns (does not exit the program), most of the MBX functions return an error.

XmbufQueryExtension

This function determines whether a display supports the MBX extension.

```
Bool
XmbufQueryExtension(display,mbuf_event_base,mbuf_error_base)
Display *display;
int *mbuf_event_base; /* RETURN */
int *mbuf_error_base; /* RETURN */
```

Arguments

display
Specifies the connection to the X server.

mbuf_event_base
Returns the first event code used by the extension. An `XEvent` with a type field equal to **mbuf_event_base* + `MultibufferClobberNotify` is a `ClobberNotify` event. An `XEvent` with a type field equal to **mbuf_event_base* + `MultibufferUpdateNotify` is an `UpdateNotify` event.

mbuf_error_base
Returns the first error code used by the extension. An `XErrorEvent` with an `error_code` field equal to `*mbuf_error_base` + `MultibufferBadBuffer` is a `BadBuffer` error.

Description

If the given display supports the MBX extension, `XmbufQueryExtension` fills in `*mbuf_event_base` and `*mbuf_error_base` and returns `True`, else it returns `False` without changing `*mbuf_event_base` and `*mbuf_error_base`.

XmbufGetVersion

This function retrieves the major and minor version numbers of the MBX extension.

```
Status
XmbufGetVersion(display,major_version,minor_version)
Display *display;
int *major_version; /* RETURN */
int *minor_version; /* RETURN */
```

Arguments

display
Specifies the connection to the X server.

major_version
Returns the major version number of the extension.

minor_version
Returns the minor version number of the extension.

Description

If no error occurs, `XmbufGetVersion` fills in `*major_version` and `*minor_version` with the version of the extension supported by the display and returns non–zero, else it returns zero without changing `*major_version` and `*minor_version`.

Protocol

Issues a `GetBufferVersion` request.

XmbufCreateBuffers

This function requests a specified number of image buffers to be associated with a window.

```
int
XmbufCreateBuffers(display,window,count,update_action,
                   update_hint,buffers)
Display *display;
Window window;
int count;
int update_action, update_hint;
Multibuffer *buffers; /* RETURN */
```

Arguments

display
Specifies the connection to the X server.

window
Specifies the window with which the buffers should be associated.

count
Specifies the number of buffers.

update_action
Specifies the update action to be applied to the buffers. See “Update Action Constants” on page 109 for allowable values.

update_hint
Specifies the update hint for the buffers. See “Update Hint Constants” on page 109 for allowable values.

buffers
Must be a pointer to enough memory to hold *count* multibuffers. Returns the multibuffer IDs that were created.

Description

`XmbufCreateBuffers` attempts to create `count` buffers associated with the given window. The requested number of buffers may not be able to be satisfied and less than *count* buffers may actually be allocated. The number of buffers actually allocated is returned. This many multibuffer IDs will be returned in `*buffers`. If an error occurs, `XmbufCreateBuffers` returns zero and leaves `*buffers` undefined.

`buffers` must always be large enough to hold at least `count` multibuffers.

The buffers are assigned the given `update_action` and `update_hint`.

No `BadAlloc` errors are ever generated due to lack of buffers because, in the worst case, `buffers[0]` can always be associated with the existing displayed image buffer of the window. In this case, one buffer still can be returned. However, `BadAlloc` may still be returned if temporary memory needed to execute the request cannot be allocated.

Diagnostics

BadWindow
`window` does not name a defined window.

BadValue
`update_action` or `update_hint` is invalid.

BadIDChoice
At least one of the multibuffer IDs in `buffers` is an invalid resource ID.

BadMatch
`window` is an `InputOnly` window.

BadAlloc
The system failed to allocate the necessary temporary memory to execute the request.

Protocol

Issues a `CreateImageBuffers` request.

XmbufDestroyBuffers

This function frees the window's associated image buffers.

```
void
XmbufDestroyBuffers(display,window)
Display *display;
Window window;
```

Arguments

display
Specifies the connection to the X server.

window
Specifies the window whose buffers are to be destroyed.

Description

Destroys the image buffers associated with the `window`.

Diagnostics

BadWindow
`window` does not name a defined `Window`.

Protocol

Issues a `DestroyImageBuffers` request.

B

XmbufDisplayBuffers

This function tells the system which image buffers are visible in the given windows.

```
void
XmbufDisplayBuffers(display,count,buffers,min_delay,max_delay)
Display *display;
int count;
Multibuffer *buffers;
int min_delay, max_delay;
```

Arguments

display
Specifies the connection to the X server.

count
Specifies the number of multibuffer IDs pointed to by buffers.

buffers
Specifies the `Multibuffers` selected for display in their associated windows.

min_delay
Specifies the minimum number of milliseconds that must elapse since the last time a `DisplayImageBuffers` was executed on a window.

max_delay
Specifies an additional delay beyond `min_delay` that the server is allowed to wait to complete the `DisplayImageBuffers` request.

Description

If no error occurs, `XmbufDisplayBuffers` displays the indicated buffers in their associated windows within the given time constraints.

Diagnostics

BadBuffer
At least one of the `Multibuffers` in `buffers` does not name a defined Buffer.

BadMatch
Two or more `Multibuffers` associated with the same window were specified in `buffers`.

BadAlloc
The system failed to allocate the necessary temporary memory to execute the request.

Protocol

Issues a `DisplayImageBuffers` request.

XmbufGetWindowAttributes

This function retrieves a window's multi-buffering attribute values.

```
Status
XmbufGetWindowAttributes(display,window,attributes)
Display *display;
Window window;
XmbufWindowAttributes *attributes; /* RETURN */
```

Arguments

display
Specifies the connection to the X server.

window
Specifies the window whose multibuffer attributes are to be retrieved.

attributes
Returns the specified window's multibuffer attributes.

Description

If no error occurs, `XmbufGetWindowAttributes` returns non–zero and stores the window's multibuffer attributes in the `XmbufWindowAttributes` structure. To free the buffers list in the attributes structure, use `XFree`. If an error occurs, `XmbufGetWindowAttributes` returns zero and leaves *attributes* unchanged.

Structures

```
typedef struct {
    int displayed_index;/* which buffer is being displayed */
    int update_action;/* see Update action constants */
    int update_hint;/* see Update hint constants */
    int window_mode;/* see Window mode constants */
    int nbuffers;/* number of buffers in following list */
    Multibuffer *buffers;/* buffer IDs associated with this window */
} XmbufWindowAttributes;
```

Diagnostics

BadWindow
`window` does not name a defined `Window`.

BadAccess
`window` is not multi–buffered.

BadValue
Not currently generated.

BadAlloc
The system failed to allocate the necessary temporary memory to execute the request.

Protocol

Issues a `GetMultiBufferAttributes` request.

XmbufChangeWindowAttributes

This function modifies a window's multi-buffering attribute values.

```
void
XmbufChangeWindowAttributes(display,window,valuemask,values)
Display *display;
Window window;
unsigned long valuemask;
XmbufSetWindowAttributes *values;
```

Arguments

display
Specifies the connection to the X server.

window
Specifies the window whose multibuffer attributes are to be changed.

valuemask
Specifies which attributes are to be changed using information in the specified attributes structure. The only value currently defined for this is `MultibufferWindowUpdateHint`.

values
Specifies any values as indicated by `valuemask`.

Description

If no error occurs, `XmbufChangeWindowAttributes` sets the multi-buffering attributes that apply to all buffers associated with the given window.

Structures

```
typedef struct {
    int update_hint;       /* see Update hint constants above */
} XmbufSetWindowAttributes;
```

Diagnostics

BadWindow
`window` does not name a defined `Window`.

BadMatch
`window` is not multi–buffered.

BadValue
`update_hint` or `valuemask` is invalid.

Protocol

Issues a `SetMultiBufferAttributes` request.

XmbufGetBufferAttributes

This function retrieves an individual image buffer's attributes.

```
Status
XmbufGetBufferAttributes(display,buffer,attributes)
Display *display;
Multibuffer buffer;
XmbufBufferAttributes *attributes; /* RETURN */
```

Arguments

display
Specifies the connection to the X server.

buffer
Specifies the buffer whose attributes are to be retrieved.

attributes
Returns the per–buffer attributes for the specified buffer.

Descriptions

If no error occurs, `XmbufGetBufferAttributes` fills in the `attributes` structure with values of the per–buffer attributes for the indicated buffer and returns non–zero, else it returns zero and leaves `attributes` unchanged.

Structures

```
typedef struct {
    Window window;     /* which window this buffer belongs to */
    unsigned long event_mask;/* events selected for this buffer */
    int buffer_index;      /* which buffer is this */
    int side;              /* see Window side constants above */
} XmbufBufferAttributes;
```

Diagnostics

BadBuffer
`buffer` does not name a defined `Buffer`.

BadValue
Not currently generated.

Protocol

Issues a `GetBufferAttributes` request.

XmbufChangeBufferAttributes

This function modifies an individual image buffer's attribute values.

```
void
XmbufChangeBufferAttributes(display,buffer,valuemask,values)
Display *display;
Multibuffer buffer;
unsigned long valuemask;
XmbufSetBufferAttributes *values;
```

Arguments

display
Specifies the connection to the X server.

buffer
Specifies the buffer whose attributes are to be changed.

valuemask
Specifies the specific buffer attributes to be changed. The only value currently defined for this is `MultibufferBufferEventMask`.

values
Specifies any values as indicated by `valuemask`.

Description

If no error occurs, `XmbufChangeBufferAttributes` sets the attributes for the indicated buffer.

Structures

```
typedef struct {
    unsigned long event_mask;/* see Event mask constants above */
} XmbufSetBufferAttributes;
```

Diagnostics

BadBuffer
`buffer` does not name a defined `Buffer`.

BadValue
`valuemask` or `values->event_mask` is invalid.

Protocol

Issues a `SetBufferAttributes` request.

XmbufGetScreenInfo

This function retrieves information about the visuals on a screen that support multi-buffering.

```
Status
XmbufGetScreenInfo(display,drawable,nmono,mono_info,nstereo,
     stereo_info)
Display *display;
Drawable drawable;
int *nmono;                     /* RETURN */
XmbufBufferInfo **mono_info;  /* RETURN */
int *nstereo;                   /* RETURN */
XmbufBufferInfo **stereo_info;/* RETURN */
```

Arguments

display
Specifies the connection to the X server.

drawable
Specifies a drawable on the screen whose buffer information is to be retrieved.

nmono
Returns the number of entries in the `mono_info` list.

mono_info
Returns a list of structures describing which monoscopic visuals are multi–buffered.

nstereo
Returns the number of entries in the `stereo_info` list.

Note – The stereo features of MBX are not supported in Solaris, so the value of `nstereo` is always `0` for all screens.

---

stereo_info
Returns a list of structures describing which stereoscopic visuals are multi–buffered.

Description

If no error occurs, `XmbufGetScreenInfo` returns non–zero and gets the parameters defining the characteristics of the multi-buffered windows that may be created on the screen of the given drawable. If `*nmono` is greater than zero, then `*mono_info` is set to the address of an array of `XmbufBufferInfo` structures describing the various visuals and depths that may be used to create multi-buffered windows. Otherwise, `*mono_info` is set to `NULL`. To release the storage returned in `*mono_info`, use `XFree`. If an error occurs, `XmbufGetScreenInfo` returns zero and leaves `*nmono`, `*mono_info`, `*nstereo`, and `*stereo_info` unchanged.

Structures

```
typedef struct {
    VisualID visualid;     /* visual usable at specified depth */
    int max_buffers;       /* max. num. of bufs for this visual */
    int depth;             /* depth of buffers creatable */
} XmbufBufferInfo;
```

Diagnostics

BadDrawable
`drawable` does not name a defined `Drawable`.

BadAlloc
The system failed to allocate the necessary temporary memory to execute the request.

Protocol

Issues a `GetBufferInfo` request.

B