X Synchronization Extension Library

X Consortium Standard

Tim Glauert, Olivetti Research/MultiWorks
Dave Carver
Digital Equipment Corporation, MIT/Project Athena
Jim Gettys
Digital Equipment Corporation, Cambridge Research Laboratory
David Wiggins
X Consortium, Inc.
# Table of Contents

1. Synchronization Protocol ................................................................. 1  
   Description ..................................................................................... 1  
2. C Language Binding ......................................................................... 2  
   C Functions ..................................................................................... 2  
   C Macros/Functions ......................................................................... 4  
   Events .............................................................................................. 5  
   Errors .............................................................................................. 6
Chapter 1. Synchronization Protocol

The core X protocol makes no guarantees about the relative order of execution of requests for different clients. This means that any synchronization between clients must be done at the client level in an operating system-dependent and network-dependent manner. Even if there was an accepted standard for such synchronization, the use of a network introduces unpredictable delays between the synchronization of the clients and the delivery of the resulting requests to the X server.

The core X protocol also makes no guarantees about the time at which requests are executed, which means that all clients with real-time constraints must implement their timing on the host computer. Any such timings are subject to error introduced by delays within the operating system and network and are inefficient because of the need for round-trip requests that keep the client and server synchronized.

The synchronization extension provides primitives that allow synchronization between clients to take place entirely within the X server. This removes any error introduced by the network and makes it possible to synchronize clients on different hosts running different operating systems. This is important for multimedia applications, where audio, video, and graphics data streams are being synchronized. The extension also provides internal timers within the X server to which client requests can be synchronized. This allows simple animation applications to be implemented without any round-trip requests and makes best use of buffering within the client, network, and server.

Description

The mechanism used by this extension for synchronization within the X server is to block the processing of requests from a client until a specific synchronization condition occurs. When the condition occurs, the client is released and processing of requests continues. Multiple clients may block on the same condition to give inter-client synchronization. Alternatively, a single client may block on a condition such as an animation frame marker.

The extension adds Counter and Alarm to the set of resources managed by the server. A counter has a 64-bit integer value that may be increased or decreased by client requests or by the server internally. A client can block by sending an Await request that waits until one of a set of synchronization conditions, called TRIGGERs, becomes TRUE.

The CreateCounter request allows a client to create a Counter that can be changed by explicit SetCounter and ChangeCounter requests. These can be used to implement synchronization between different clients.

There are some counters, called System Counters, that are changed by the server internally rather than by client requests. The effect of any change to a system counter is not visible until the server has finished processing the current request. In other words, system counters are apparently updated in the gaps between the execution of requests rather than during the actual execution of a request. The extension provides a system counter that advances with the server time as defined by the core protocol, and it may also provide counters that advance with the real-world time or that change each time the CRT screen is refreshed. Other extensions may provide their own extension-specific system counters.

The extension provides an Alarm mechanism that allows clients to receive an event on a regular basis when a particular counter is changed.
Chapter 2. C Language Binding

The C routines provide direct access to the protocol and add no additional semantics.

The include file for this extension is `<X11/extensions/sync.h>`. Most of the names in the language binding are derived from the protocol names by prepending XSync to the protocol name and changing the capitalization.

C Functions

Most of the following functions generate SYNC protocol requests.

```c
Status XSyncQueryExtension( *dpy, *event_base_return, *error_base_return);
```

If dpy supports the SYNC extension, XSyncQueryExtension returns True, sets *event_base_return to the event number for the first SYNC event, and sets *error_base_return to the error number for the first SYNC error. If dpy does not support the SYNC extension, it returns False.

```c
Status XSyncInitialize( *dpy, *major_version_return, *minor_version_return);
```

XSyncInitialize sets *major_version_return and *minor_version_return to the major/minor SYNC protocol version supported by the server. If the XSync library is compatible with the version returned by the server, this function returns True. If dpy does not support the SYNC extension, or if there was an error during communication with the server, or if the server and library protocol versions are incompatible, this function returns False. The only XSync function that may be called before this function is XSyncQueryExtension. If a client violates this rule, the effects of all XSync calls that it makes are undefined.

```c
XSyncSystemCounter *XSyncListSystemCounters( *dpy, *n_counters_return);
```

XSyncListSystemCounters returns a pointer to an array of system counters supported by the display and sets *n_counters_return to the number of counters in the array. The array should be freed with XSyncFreeSystemCounterList. If dpy does not support the SYNC extension, or if there was an error during communication with the server, or if the server does not support any system counters, this function returns NULL.

XSyncSystemCounter has the following fields:

- char * name; /* null-terminated name of system counter */
- XSyncCounter counter; /* counter id of this system counter */
- XSyncValue resolution; /* resolution of this system counter */

```c
void XSyncFreeSystemCounterList( *list);
```

XSyncFreeSystemCounterList frees the memory associated with the system counter list returned by XSyncListSystemCounters.

```c
XSyncCounter XSyncCreateCounter( *dpy, initial_value);
```

XSyncCreateCounter creates a counter on the dpy with the given initial value and returns the counter ID. It returns None if dpy does not support the SYNC extension.
C Language Binding

```c
Status XSyncSetCounter( *dpy, counter, value);
XSyncSetCounter sets counter to value. It returns False if dpy does not support the SYNC extension; otherwise, it returns True.

Status XSyncChangeCounter( *dpy, counter, value);
XSyncChangeCounter adds value to counter. It returns False if dpy does not support the SYNC extension; otherwise, it returns True.

Status XSyncDestroyCounter( *dpy, counter);
XSyncDestroyCounter destroys counter. It returns False if dpy does not support the SYNC extension; otherwise, it returns True.

Status XSyncQueryCounter( *dpy, counter, *value_return);
XSyncQueryCounter sets *value_return to the current value of counter. It returns False if there was an error during communication with the server or if dpy does not support the SYNC extension; otherwise, it returns True.

Status XSyncAwait( *dpy, *wait_list, n_conditions);
XSyncAwait awaits on the conditions in wait_list. The n_conditions is the number of wait conditions in wait_list. It returns False if dpy does not support the SYNC extension; otherwise, it returns True. The await is processed asynchronously by the server; this function always returns immediately after issuing the request.

XSyncWaitCondition has the following fields:

- XSyncCounter trigger.counter; /*counter to trigger on */
- XSyncValueType trigger.value_type; /*absolute/relative */
- XSyncValue trigger.wait_value; /*value to compare counter to */
- XSyncTestType trigger.test_type; /*pos/neg comparison/transition */
- XSyncValue event_threshold; /*send event if past threshold */

XSyncValueType can be either XSyncAbsolute or XSyncRelative.

XSyncTestType can be one of XSyncPositiveTransition, XSyncNegativeTransition, XSyncPositiveComparison, or XSyncNegativeComparison.

XSyncAlarm XSyncCreateAlarm( *dpy, values_mask, *values`);
XSyncCreateAlarm creates an alarm and returns the alarm ID. It returns None if the display does not support the SYNC extension. The values_mask and values specify the alarm attributes.

XSyncAlarmAttributes has the following fields. The attribute_mask column specifies the symbol that the caller should OR into values_mask to indicate that the value for the corresponding attribute was actually supplied. Default values are used for all attributes that do not have their attribute_mask OR’ed into values_mask. See the protocol description for CreateAlarm for the defaults.

<table>
<thead>
<tr>
<th>type</th>
<th>field name</th>
<th>attribute_mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSyncCounter</td>
<td>trigger.counter;</td>
<td>XSyncCACounter</td>
</tr>
<tr>
<td>XSyncValueType</td>
<td>trigger.value_type;</td>
<td>XSyncCAValueType</td>
</tr>
<tr>
<td>XSyncValue</td>
<td>trigger.wait_value;</td>
<td>XSyncCAValue</td>
</tr>
</tbody>
</table>
```

XSyncTestType trigger.test_type; XSyncCATestType
XSyncValue delta; XSyncCADelta
Bool events; XSyncCAEvents
XSyncAlarmState state; client cannot set this

Status XSyncDestroyAlarm( *dpy, alarm);

XSyncDestroyAlarm destroys alarm. It returns False if dpy does not support the SYNC extension; otherwise, it returns True.

Status XSyncQueryAlarm( *dpy, alarm, *values_return);

XSyncQueryAlarm sets *values_return to the alarm’s attributes. It returns False if there was an error during communication with the server or if dpy does not support the SYNC extension; otherwise, it returns True.

Status XSyncChangeAlarm( *dpy, alarm, values_mask, *values);

XSyncChangeAlarm changes alarm’s attributes. The attributes to change are specified as in XSyncCreateAlarm. It returns False if dpy does not support the SYNC extension; otherwise, it returns True.

Status XSyncSetPriority( *dpy, client_resource_id, priority);

XSyncSetPriority sets the priority of the client owning client_resource_id to priority. If client_resource_id is None, it sets the caller’s priority. It returns False if dpy does not support the SYNC extension; otherwise, it returns True.

Status XSyncGetPriority( *dpy, client_resource_id, *return_priority);

XSyncGetPriority sets *return_priority to the priority of the client owning client_resource_id. If client_resource_id is None, it sets *return_priority to the caller’s priority. It returns False if there was an error during communication with the server or if dpy does not support the SYNC extension; otherwise, it returns True.

C Macros/Functions

The following procedures manipulate 64-bit values. They are defined both as macros and as functions. By default, the macro form is used. To use the function form, #undef the macro name to uncover the function.

void XSyncIntToValue( *pv, i);

Converts i to an XSyncValue and stores it in *pv. Performs sign extension (*pv will have the same sign as i.)

void XSyncIntsToValue( *pv, low, high);

Stores low in the low 32 bits of *pv and high in the high 32 bits of *pv.

Bool XSyncValueGreaterThan( a, b);

Returns True if a is greater than b, else returns False.

Bool XSyncValueLessThan( a, b);

Returns True if a is less than b, else returns False.
Bool XSyncValueGreaterOrEqual( a, b);

Returns True if a is greater than or equal to b, else returns False.

Bool XSyncValueLessOrEqual( a, b);

Returns True if a is less than or equal to b, else returns False.

Bool XSyncValueEqual( a, b);

Returns True if a is equal to b, else returns False.

Bool XSyncValueIsNegative( v);

Returns True if v is negative, else returns False.

Bool XSyncValueIsZero( v);

Returns True if v is zero, else returns False.

Bool XSyncValueIsPositive( v);

Returns True if v is positive, else returns False.

unsigned int XSyncValueLow32( v);

Returns the low 32 bits of v.

unsigned int XSyncValueHigh32( v);

Returns the high 32 bits of v.

void XSyncValueAdd( *presult, a, b, *poverflow);

Adds a to b and stores the result in *presult. If the result could not fit in 64 bits, *poverflow is set to True, else it is set to False.

void XSyncValueSubtract( *presult, a, b, *poverflow);

Subtracts b from a and stores the result in *presult. If the result could not fit in 64 bits, *poverflow is set to True, else it is set to False.

void XSyncMaxValue( *pv);

Sets *pv to the maximum value expressible in 64 bits.

void XSyncMinValue( *pv);

Sets *pv to the minimum value expressible in 64 bits.

Events

Let event_base be the value event base return as defined in the function XSyncQueryExtension.

An XSyncCounterNotifyEvent's type field has the value event_base + XSyncCounterNotify. The fields of this structure are:
int type; /* event_base + XSyncCounterNotify */
unsigned long serial; /* number of last request processed by server */
Bool send event; /* true if this came from a SendEvent request */
Display * display; /* Display the event was read from */
XSyncCounter counter; /* counter involved in await */
XSyncValue wait_value; /* value being waited for */
XSyncValue counter_value; /* counter value when this event was sent */
Time time; /* milliseconds */
int count; /* how many more events to come */
Bool destroyed; /* True if counter was destroyed */

An XSyncAlarmNotifyEvent’s type field has the value event_base + XSyncAlarmNotify. The fields of this structure are:

int type; /* event_base + XSyncAlarmNotify */
unsigned long serial; /* number of last request processed by server */
Bool send_event; /* true if this came from a SendEvent request */
Display * display; /* Display the event was read from */
XSyncAlarm alarm; /* alarm that triggered */
XSyncValue counter_value; /* value that triggered the alarm */
XSyncValue alarm_value; /* test value of trigger in alarm */
Time time; /* milliseconds */
XSyncAlarmState state; /* new state of alarm */

Errors

Let error_base be the value error_base_return as defined in the function XSyncQueryExtension.

An XSyncAlarmError’s error_code field has XSyncBadAlarm. The fields of this structure are:

int type
Display * display; /* Display the event was read from */
XSyncCounter counter; /* resource id */
unsigned long serial; /* serial number of failed request */
unsigned char error_code; /* error_base + XSyncBadAlarm */
unsigned char request_code; /* Major op-code of failed request */
unsigned char minor_code; /* Minor op-code of failed request */

An XSyncCounterError’s error code field has the value error_base + XSyncBadCounter. The fields of this structure are:

int type
Display * display; /* Display the event was read from */
XSyncCounter counter; /* resource id */
unsigned long serial; /* serial number of failed request */
unsigned char error_code; /* error_base + XSyncBadCounter */
unsigned char request_code; /* Major op-code of failed request */
unsigned char minor_code; /* Minor op-code of failed request */