NAME

XScreenSaver - X11 Screen Saver extension client library

SYNOPSIS

```
#include <X11/extensions/scrnsaver.h>

typedef struct {
    Window window;              /* screen saver window */
    int state;                  /* ScreenSaver{Off,On,Disabled} */
    int kind;                   /* ScreenSaver{Blanked,Internal,External} */
    unsigned long til_or_since; /* milliseconds */
    unsigned long idle;         /* milliseconds */
    unsigned long eventMask;    /* events */
} XScreenSaverInfo;

typedef struct {
    int type;               /* of event */
    unsigned long serial;   /* # 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 */
    Window window;          /* screen saver window */
    Window root;            /* root window of event screen */
    int state;              /* ScreenSaver{Off,On,Cycle} */
    int kind;               /* ScreenSaver{Blanked,Internal,External} */
    Bool forced;            /* extents of new region */
    Time time;              /* event timestamp */
} XScreenSaverNotifyEvent;

Bool XScreenSaverQueryExtension(Display *dpy, int *event_base_return, int *error_base_return);

Status XScreenSaverQueryVersion(Display *dpy, int *major_version_return, int
    *minor_version_return);

XScreenSaverInfo *XScreenSaverAllocInfo(void);

Status XScreenSaverQueryInfo(Display *dpy, Drawable drawable, XScreenSaverInfo *saver_info);

void XScreenSaverSelectInput(Display *dpy, Drawable drawable, unsigned long mask);

void XScreenSaverSetAttributes(Display *dpy, Drawable drawable, int x, int y, unsigned int width,
```

unsigned int *height*, unsigned int *border_width*, int *depth*, unsigned int *class*, Visual **visual*, unsigned long *valuemask*, XSetWindowAttributes **attributes*);

void XScreenSaverUnsetAttributes(Display **dpy*, Drawable *drawable*);

void XScreenSaverRegister(Display **dpy*, int *screen*, XID *xid*, Atom *type*);

Status XScreenSaverUnregister(Display **dpy*, int *screen*);

Status XScreenSaverGetRegistered(Display **dpy*, int *screen*, XID **xid*, Atom **type*);

void XScreenSaverSuspend(Display **dpy*, Bool *suspend*);

DESCRIPTION

The X Window System provides support for changing the image on a display screen after a user-settable period of inactivity to avoid burning the cathode ray tube phosphors. However, no interfaces are provided for the user to control the image that is drawn. This extension allows an external ‘‘screen saver’’ client to detect when the alternate image is to be displayed and to provide the graphics.

Current X server implementations typically provide at least one form of ‘‘screen saver’’ image. Historically, this has been a copy of the X logo drawn against the root background pattern. However, many users have asked for the mechanism to allow them to write screen saver programs that provide capabilities similar to those provided by other window systems. In particular, such users often wish to be able to display corporate logos, instructions on how to reactivate the screen, and automatic screen-locking utilities. This extension provides a means for writing such clients.

Assumptions

This extension exports the notion of a special screen saver window that is mapped above all other windows on a display. This window has the *override-redirect* attribute set so that it is not subject to manipulation by the window manager. Furthermore, the X identifier for the window is never returned by **QueryTree** requests on the root window, so it is typically not visible to other clients.

XScreenSaverQueryExtension returns **True** if the *XScreenSaver* extension is available on the given display. A client must call **XScreenSaverQueryExtension** before calling any other XScreenSaver function in order to negotiate a compatible protocol version; otherwise the client will get undefined behavior (XScreenSaver may or may not work).

If the extension is supported, the event number for *ScreenSaverNotify* events is returned in the value pointed to by *event_base*. Since no additional errors are defined by this extension, the results of *error_base* are not defined.

XScreenSaverQueryVersion returns **True** if the request succeeded; the values of the major and minor protocol versions supported by the server are returned in *major_version_return* and *minor_version_return* .

XScreenSaverAllocInfo allocates and returns an **XScreenSaverInfo** structure for use in calls to **XScreenSaverQueryInfo**. All fields in the structure are initialized to zero. If insufficient memory is available, NULL is returned. The results of this routine can be released using *XFree*.

XScreenSaverQueryInfo returns information about the current state of the screen server in *saver_info* and a non-zero value is returned. If the extension is not supported, *saver_info* is not changed and 0 is returned.

The *state* field specifies whether or not the screen saver is currently active and how the *til-or-since* value should be interpreted:

Off The screen is not currently being saved; *til-or-since* specifies the number of milliseconds until the screen saver is expected to activate.

On The screen is currently being saved; *til-or-since* specifies the number of milliseconds since the screen saver activated.

Disabled
The screen saver is currently disabled; *til-or-since* is zero.

The *kind* field specifies the mechanism that either is currently being used or would have been were the screen being saved:

Blanked
The video signal to the display monitor was disabled.

Internal
A server-dependent, built-in screen saver image was displayed; either no client had set the screen saver window attributes or a different client had the server grabbed when the screen saver activated.

External
The screen saver window was mapped with attributes set by a client using the **ScreenSaverSetAttributes** request.

The *idle* field specifies the number of milliseconds since the last input was received from the user on

any of the input devices.
The *event-mask* field specifies which, if any, screen saver events this client has requested using **ScreenSaverSelectInput**.

XScreenSaverSelectInput asks that events related to the screen saver be generated for this client. If no bits are set in *event-mask*, then no events will be generated. Otherwise, any combination of the following bits may be set:

ScreenSaverNotify
If this bit is set, **ScreenSaverNotify** events are generated whenever the screen saver is activated or deactivated.

ScreenSaverCycle
If this bit is set, **ScreenSaverNotify** events are generated whenever the screen saver cycle interval passes.

XScreenSaverSetAttributes sets the attributes to be used the next time the external screen saver is activated. If another client currently has the attributes set, a BadAccess error is generated and the request is ignored.
Otherwise, the specified window attributes are checked as if they were used in a core **CreateWindow** request whose parent is the root. The *override-redirect* field is ignored as it is implicitly set to True. If the window attributes result in an error according to the rules for **CreateWindow**, the request is ignored.
Otherwise, the attributes are stored and will take effect on the next activation that occurs when the server is not grabbed by another client. Any resources specified for the *background-pixmap* or *cursor* attributes may be freed immediately. The server is free to copy the *background-pixmap* or *cursor* resources or to use them in place; therefore, the effect of changing the contents of those resources is undefined. If the specified *colormap* no longer exists when the screen saver activates, the parent's colormap is used instead. If no errors are generated by this request, any previous screen saver window attributes set by this client are released.
When the screen saver next activates and the server is not grabbed by another client, the screen saver window is created, if necessary, and set to the specified attributes and events are generated as usual. The colormap associated with the screen saver window is installed. Finally, the screen saver window is mapped.
The window remains mapped and at the top of the stacking order until the screen saver is deactivated in response to activity on any of the user input devices, a **ForceScreenSaver** request with a value of Reset, or any request that would cause the window to be unmapped.
If the screen saver activates while the server is grabbed by another client, the internal saver mechanism is used. The **ForceScreenSaver** request may be used with a value of Active to deactivate the internal saver and activate the external saver.
If the screen saver client's connection to the server is broken while the screen saver is activated and the

client's close down mode has not been RetainPermanent or RetainTemporary, the current screen saver is deactivated and the internal screen saver is immediately activated.
When the screen saver deactivates, the screen saver window's colormap is uninstalled and the window is unmapped (except as described below). The screen saver XID is disassociated with the window and the server may, but is not required to, destroy the window along with any children.
When the screen saver is being deactivated and then immediately reactivated (such as when switching screen savers), the server may leave the screen saver window mapped (typically to avoid generating exposures).

XScreenSaverUnsetAttributes instructs the server to discard any previous screen saver window attributes set by this client.

XScreenSaverRegister stores the given *XID* in the **_SCREEN_SAVER_ID** property (of the given *type*) on the root window of the specified *screen*. It returns zero if an error is encountered and the property is not changed, otherwise it returns non-zero.

XScreenSaverUnregister removes any **_SCREEN_SAVER_ID** from the root window of the specified *screen*. It returns zero if an error is encountered and the property is changed, otherwise it returns non-zero.

XScreenSaverGetRegistered returns the *XID* and *type* stored in the **_SCREEN_SAVER_ID** property on the root window of the specified *screen*. It returns zero if an error is encountered or if the property does not exist or is not of the correct format; otherwise it returns non-zero.

XScreenSaverSuspend temporarily suspends the screensaver and DPMS timer if *suspend* is 'True', and restarts the timer if *suspend* is 'False'.
This function should be used by applications that don't want the screensaver or DPMS to become activated while they're for example in the process of playing a media sequence, or are otherwise continuously presenting visual information to the user while in a non-interactive state. This function is not intended to be called by an external screensaver application.
If **XScreenSaverSuspend** is called multiple times with *suspend* set to 'True', it must be called an equal number of times with *suspend* set to 'False' in order for the screensaver timer to be restarted. This request has no affect if a client tries to resume the screensaver without first having suspended it. **XScreenSaverSuspend** can thus not be used by one client to resume the screensaver if it's been suspended by another client.
If a client that has suspended the screensaver becomes disconnected from the X server, the screensaver timer will automatically be restarted, unless it's still suspended by another client. Suspending the screensaver timer doesn't prevent the screensaver from being forcibly activated with the **ForceScreenSaver** request, or a DPMS mode from being set with the **DPMSForceLevel** request. **XScreenSaverSuspend** also doesn't deactivate the screensaver or DPMS if either is active at the time

the request to suspend them is received by the X server. But once they've been deactivated, they won't automatically be activated again, until the client has canceled the suspension.

ERRORS

XScreenSaverSelectInput, XScreenSaverQueryInfo, XScreenSaverSetAttributes and **XScreenSaverUnsetAttributes** will generate a *BadDrawable* error if *drawable* is not a valid drawable identifier. If any undefined bits are set in *event-mask*, a BadValue error is generated by **XScreenSaverSelectInput .**

AVAILABILITY

XScreenSaverSuspend is available in version 1.1 and later versions of the X Screen Saver Extension. Version 1.1 was first released with X11R7.1.

SEE ALSO

X(7)

AUTHORS

Jim Fulton and Keith Packard.

STABILITY

This API is considered as experimental. The Xss library major revision may be incremented whenever incompatible changes are done to the API without notice. Use with care.