X11 Screen Saver Extension

MIT X Consortium Proposed Standard

Jim Fulton, Network Computing Devices, Inc
Keith Packard, X Consortium, Laboratory for Computer Science, Massachusetts Institute of Technology

X11 Screen Saver Extension: MIT X Consortium Proposed Standard

by Jim Fulton and Keith Packard

Version 1.0

Table of Contents

Chapter 1. Introduction

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.

Chapter 2. 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.

Chapter 3. Overview

The core `SetScreenSaver` request can be used to set the length of time without activity on any input devices after which the screen saver should "activate" and alter the image on the screen. This image periodically "cycles" to reduce the length of time that any particular pixel is illuminated. Finally, the screen saver is "deactivated" in response to activity on any of the input devices or particular X requests.

Screen saving is typically done by disabling video output to the display tube or by drawing a changing pattern onto the display. If the server chooses the latter approach, a window with a special identifier is created and mapped at the top of the stacking order where it remains until the screen saver deactivates. At this time, the window is unmapped and is not accessible to any client requests.

The server's default mechanism is refered to as the *internal* screen saver. An *external* screen saver client requires a means of determining the window id for the screen saver window and setting the attributes (e.g. size, location, visual, colormap) to be used when the window is mapped. These requirements form the basis of this extension.

Chapter 4. Issues

This extension raises several interesting issues. First is the question of what should be done if some other client has the server grabbed when the screen saver is supposed to activate? This commonly occurs with window managers that automatically ask the user to position a window when it is first mapped by grabbing the server and drawing XORed lines on the root window.

Second, a screen saver program must control the actual RGB values sent to the display tube to ensure that the values change periodically to avoid phosphor burn in. Thus, the client must have a known colormap installed whenever the screen saver window is displayed. To prevent screen flashing, the visual type of the screen saver window should also be controlable.

Third, some implementations may wish to destroy the screen saver window when it is not mapped so that it need not be avoided during event delivery. Thus, screen saver clients may find that the requests that reference the screen saver window may fail when the window is not displayed.

Chapter 5. Protocol

The Screen Saver extension is as follows:

Types

In adition to the comon types described in the core protocol, the following type is used in the request and event definitions in subsequent sections.

Name	**Value**
SCREENSAVEREVENT	**ScreenSaverNotify**, **ScreenSaverCycle**

Errors

The Screen Saver extension adds no errors beyond the core protocol.

Requests

The Screen Saver extension adds the following requests:

ScreenSaverQueryVersion
    client-major-version: CARD8
    client-minor-version: CARD8
->
    server-major-version: CARD8
    server-minor-version: CARD8

This request allows the client and server to determine which version of the protocol should be used. The client sends the version that it prefers; if the server understands that version, it returns the same values and interprets subsequent requests for this extension according to the specified version. Otherwise, the server returns the closest version of the protocol that it can support and interprets subsequent requests according to that version. This document describes major version 1, minor version 0; the major and minor revision numbers should only be incremented in response to incompatible and compatible changes, respectively.

ScreenSaverQueryInfo
drawable DRAWABLE

saver-window: WINDOW
state: {**Disabled**, **Off**, **On**}
kind: {**Blanked**, **Internal**, **External**}
til-or-since: CARD32
idle: CARD32
event-mask: SETofSCREENSAVEREVENT

Errors: **Drawable**

This request returns information about the state of the screen saver on the screen associated with *drawable*. The *saver-window* is the XID that is associated with the screen saver window. This window is not guaranteed to exist except when external screen saver is active. Although it is a child of the root, this window is not returned by `QueryTree` requests on the root. Whenever this window is mapped, it is always above any of its siblings in the stacking order. XXX - TranslateCoords?

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

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`.

If *drawable* is not a valid drawable identifier, a Drawable error is returned and the request is ignored.

ScreenSaverSelectInput
drawable: DRAWABLE
event-mask: SETofSCREENSAVEREVENT

Errors: **Drawable**, **Match**

This request specifies which Screen Saver extension events on the screen associated with *drawable* should 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.

If *drawable* is not a valid drawable identifier, a Drawable error is returned. If any undefined bits are set in *event-mask*, a Value error is returned. If an error is returned, the request is ignored.

ScreenSaverSetAttributes

drawable: DRAWABLE
class:
{**InputOutput**, **InputOnly**, **CopyFromParent**}
depth: CARD8
visual: VISUALID or **CopyFromParent**
x, y: INT16
width, height, border-width: CARD16
value-mask: BITMASK
value-list: LISTofVALUE

Access, **Window**, **Pixmap**, **Colormap**, **Cursor**, **Match**, **Value**, **Alloc**

This request sets the attributes that this client would like to see used in creating the screen saver window on the screen associated with *drawable*. If another client currently has the attributes set, an Access 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).

ScreenSaverUnsetAttributes

drawble: **DRAWABLE**

Errors: **Drawable**

This request notifies the server that this client no longer wishes to control the screen saver window. Any screen saver attributes set by this client and any descendents of the screen saver window created by this client should be released immediately if the screen saver is not active, else upon deactivation.

This request is ignored if the client has not previously set the screen saver window attributes.

Events

The Screen Saver extension adds one event:

ScreenSaverNotify

root: WINDOW
window: WINDOW
state: {**Off**, **On**, **Cycle**}
kind: { **Blanked**, **Internal** , **External** }
forced: BOOL
time: TIMESTAMP

This event is delivered to clients that have requested ScreenSaverNotify events using the `ScreenSaverSelectInput` request whenever the screen saver activates or deactivates.

The *root* field specifies root window of the screen for which the event was generated. The *window* field specifies the value that is returned by `ScreenSaverQueryInfo` as the identifier for the screen saver window. This window is not required to exist if the external screen saver is not active.

The *state* field specifies the cause of the event:

Off	The screen saver deactivated; this event is sent if the client has set the ScreenSaverNotify bit in its event mask.
On	The screen saver activated. This event is sent if the client has set the ScreenSaverNotify bit in its event mask.
Cycle	The cycle interval passed and the client is expected to change the image on the screen. This event is sent if the client has set the ScreenSaverCycle bit in its event mask.

If *state* is set to **On** or **Off** then *forced* indicates whether or not activation or deactivation was caused by a core `ForceScreenSaver` request; otherwise, *forced* is set to False.

The *kind* field specifies mechanism that was used to save the screen when the screen saver was activated, as described in `ScreenSaverQueryInfo`.

The *time* field indicates the server time when the event was generated.

Chapter 6. Encoding

Please refer to the X11 Protocol Encoding document as this document uses conventions established there.

The name of this extension is "SCREEN-SAVER".

Common Types

```
SETofSCREENSAVEREVENT
     #x00000001     ScreenSaverNotifyMask
     #x00000002     ScreenSaverCycleMask
```

Requests

ScreenSaverQueryVersion

```
1     CARD8                    screen saver opcode
1     0                        minor opcode
2     2                        request length
1     CARD8                    client major version
1     CARD8                    client minor version
2                              unused
->
1     1                        Reply
1                              unused
2     CARD16                   sequence number
4     0                        reply length
1     CARD8                    server major version
1     CARD8                    server minor version
22                             unused
```

ScreenSaverQueryInfo

```
1     CARD8                    screen saver opcode
1     1                        minor opcode
2     2                        request length
4     DRAWABLE                 drawable associated with screen
->
1     1                        Reply
1     CARD8                    state
      0           Off
      1           On
      3           Disabled
2     CARD16                   sequence number
4     0                        reply length
4     WINDOW                   saver window
4     CARD32                   milliseconds until saver or since saver
4     CARD32                   milliseconds since last user device input
4     SETofSCREENSAVEREVENT    event mask
1     CARD8                    kind
```

```
     0          Blanked
     1          Internal
     2          External
10              unused
```

ScreenSaverSelectInput

```
1     CARD8                     screen saver opcode
1     2                         minor opcode
2     3                         request length
4     DRAWABLE                  drawable associated with screen
4     SETofSCREENSAVEREVENT     event mask
```

ScreenSaverSetAttributes

```
1     CARD8                     screen saver opcode
1     3                         minor opcode
2     6+n                       request length
4     DRAWABLE                  drawable associated with screen
2     INT16                     x
2     INT16                     y
2     CARD16                    width
2     CARD16                    height
2     CARD16                    border-width
1                               class
      0          CopyFromParent
      1          InputOutput
      2          InputOnly
1     CARD8                     depth
4     VISUALID                  visual
      0          CopyFromParent
4     BITMASK                   value-mask (has n bits set to 1)
      encodings are the same as for core CreateWindow
4n    LISTofVALUE               value-list
      encodings are the same as for core CreateWindow
```

ScreenSaverUnsetAttributes

```
1     CARD8                     screen saver opcode
1     4                         minor opcode
2     3                         request length
4     DRAWABLE                  drawable associated with screen
```

Events

ScreenSaverNotify

```
1     CARD8                     code assigned by core
1     CARD8                     state
      0          Off
      1          On
      2          Cycle
2     CARD16                    sequence number
4     TIMESTAMP                 time
4     WINDOW                    root
4     WINDOW                    screen saver window
```

```
1     CARD8                  kind
      0          Blanked
      1          Internal
      2          External
1     BOOL                   forced
14                           unused
```

Chapter 7. Inter-Client Communications Conventions

Screen saver clients should create at least one resource value whose identifier can be stored in a property named **_SCREEN_SAVER_ID** on the root of each screen it is managing. This property should have one 32-bit value corresponding to the resource identifier; the type of the property should indicate the type of the resource and should be one of the following: **WINDOW**, **PIXMAP**, **CURSOR**, **FONT**, or **COLORMAP**.

Chapter 8. C language binding

The C binding for this extension simply provide access to the protocol; they add no semantics beyond what is described above.

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

`Bool` **`XScreenSaverQueryExtension`**`(`*`*display`*`, `*`*event_base`*`, `*`*error_base`*`);`

This routine returns **True** if the specified *display* supports the SCREEN-SAVER extension; otherwise it returns **False**. 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.

`Status` **`XScreenSaverQueryVersion`**`(`*`*display`*`, `*`*major`*`, `*`*minor`*`);`

If the specified *display* supports the extension, the version numbers of the protocol expected by the server are returned in *major* and *minor* and a non-zero value is returned. Otherwise, the arguments are not set and 0 is returned.

XScreenSaverInfo *

XScreenSaverAllocInfo()

This routine 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*.

`Status` **`XScreenSaverQueryInfo`**`(`*`*display`*`, `*`drawable`*`, `*`*saver_info`*`);`

If the specified *display* supports the extension, information about the current state of the screen server is returned in *saver_info* and a non-zero value is returned. The `XScreenSaverInfo` structure is defined as follows:

```
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 event_mask;     /* events */
} XScreenSaverInfo;
```

See the `ScreenSaverQueryInfo` request for a description of the fields. If the extension is not supported, *saver_info* is not changed and 0 is returned.

`void` **`XScreenSaverSelectInput`**`(`*`*display`*`, `*`drawable`*`, `*`event_mask`*`);`

If the specified *display* supports the extension, this routine asks that events related to the screen saver be generated for this client. The format of the events generated is:

```
typedef struct {
    int type;               /* of event */
    unsigned long serial;   /* # of last request processed by server */
    Bool send_event;        /* true if this came frome 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;
```

See the definition of the `ScreenSaverSelectInput` request for descriptions of the allowed event masks.

`void` **`XScreenSaverSetAttributes`**`(`*`*dpy`*`,` *`drawable`*`,` *`x`*`,` *`y`*`,` *`width`*`,` *`height`*`,` *`border_width`*`,` *`depth`*`,` *`class`*`,` *`*visual`*`,` *`valuemask`*`,` *`*attributes`*`);`

If the specified *display* supports the extension, this routine sets the attributes to be used the next time the external screen saver is activated. See the definition of the `ScreenSaverSetAttributes` request for a description of each of the arguments.

`void` **`XScreenSaverUnsetAttributes`**`(`*`*display`*`,` *`drawable`*`);`

If the specified *display* supports the extension, this routine instructs the server to discard any previous screen saver window attributes set by this client.

`Status` **`XScreenSaverRegister`**`(`*`*display`*`,` *`screen`*`,` *`xid`*`,` *`type`*`);`

This routine 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.

`Status` **`XScreenSaverUnregister`**`(`*`*display`*`,` *`screen`*`);`

This routine 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.

`Status` **`XScreenSaverGetRegistered`**`(`*`*display`*`,` *`screen`*`,` *`*xid`*`,` *`*type`*`);`

This routine 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.