NAME

XSetScreenSaver, XForceScreenSaver, XActivateScreenSaver, XResetScreenSaver, XGetScreenSaver - manipulate the screen saver

SYNTAX

int XSetScreenSaver(Display **display*, int *timeout*, int *interval*, int *prefer_blanking*, int *allow_exposures*);

int XForceScreenSaver(Display **display*, int *mode*);

int XActivateScreenSaver(Display **display*);

int XResetScreenSaver(Display **display*);

int XGetScreenSaver(Display **display*, int **timeout_return*, int **interval_return*, int **prefer_blanking_return*, int **allow_exposures_return*);

ARGUMENTS

allow_exposures
: Specifies the screen save control values. You can pass **DontAllowExposures**, **AllowExposures**, or **DefaultExposures**.

allow_exposures_return
: Returns the current screen save control value (**DontAllowExposures**, **AllowExposures**, or **DefaultExposures**).

display
: Specifies the connection to the X server.

interval
: Specifies the interval, in seconds, between screen saver alterations.

interval_return
: Returns the interval between screen saver invocations.

mode
: Specifies the mode that is to be applied. You can pass **ScreenSaverActive** or **ScreenSaverReset**.

prefer_blanking
: Specifies how to enable screen blanking. You can pass **DontPreferBlanking**, **PreferBlanking**, or **DefaultBlanking**.

prefer_blanking_return

Returns the current screen blanking preference (**DontPreferBlanking**, **PreferBlanking**, or **DefaultBlanking**).

timeout Specifies the timeout, in seconds, until the screen saver turns on.

timeout_return Returns the timeout, in seconds, until the screen saver turns on.

DESCRIPTION

Timeout and interval are specified in seconds. A timeout of 0 disables the screen saver (but an activated screen saver is not deactivated), and a timeout of -1 restores the default. Other negative values generate a **BadValue** error. If the timeout value is nonzero, **XSetScreenSaver** enables the screen saver. An interval of 0 disables the random-pattern motion. Both values are limited to a 16-bit signed integer range by the wire protocol, despite the C prototype. If no input from devices (keyboard, mouse, and so on) is generated for the specified number of timeout seconds once the screen saver is enabled, the screen saver is activated.

For each screen, if blanking is preferred and the hardware supports video blanking, the screen simply goes blank. Otherwise, if either exposures are allowed or the screen can be regenerated without sending **Expose** events to clients, the screen is tiled with the root window background tile randomly re-origined each interval seconds. Otherwise, the screens' states do not change, and the screen saver is not activated. The screen saver is deactivated, and all screen states are restored at the next keyboard or pointer input or at the next call to **XForceScreenSaver** with mode **ScreenSaverReset**.

If the server-dependent screen saver method supports periodic change, the interval argument serves as a hint about how long the change period should be, and zero hints that no periodic change should be made. Examples of ways to change the screen include scrambling the colormap periodically, moving an icon image around the screen periodically, or tiling the screen with the root window background tile, randomly re-origined periodically.

XSetScreenSaver can generate a **BadValue** error.

If the specified mode is **ScreenSaverActive** and the screen saver currently is deactivated, **XForceScreenSaver** activates the screen saver even if the screen saver had been disabled with a timeout of zero. If the specified mode is **ScreenSaverReset** and the screen saver currently is enabled, **XForceScreenSaver** deactivates the screen saver if it was activated, and the activation timer is reset to its initial state (as if device input had been received).

XForceScreenSaver can generate a **BadValue** error.

The **XActivateScreenSaver** function activates the screen saver.

The **XResetScreenSaver** function resets the screen saver.

The **XGetScreenSaver** function gets the current screen saver values.

DIAGNOSTICS

BadValue Some numeric value falls outside the range of values accepted by the request. Unless a specific range is specified for an argument, the full range defined by the argument's type is accepted. Any argument defined as a set of alternatives can generate this error.

SEE ALSO

Xlib - C Language X Interface