X Display Power Management Signaling (DPMS) Extension Protocol Specification

X Project Team Standard

Rob Lembree, Digital Equipment Corporation `<lembree@zk3.dec.com>`

X Display Power Management Signaling (DPMS) Extension Protocol Specification: X Project Team Standard

by Rob Lembree

X Version 11, Release 7.7

Version 1.0

Copyright © 1996 Digital Equipment Corporation

Permission to use, copy, modify, distribute, and sell this documentation for any purpose is hereby granted without fee, provided that the above copyright notice and this permission notice appear in all copies. Digital Equipment Corporation makes no representations about the suitability for any purpose of the information in this document. This documentation is provided "as is" without express or implied warranty.

Table of Contents

1. Overview 1
2. Requests 3
3. Events and Errors 6
4. Encoding 7

Chapter 1. Overview

This extension provides X Protocol control over the VESA Display Power Management Signaling (DPMS) characteristics of video boards under control of the X Window System.[1]

Traditionally, the X Window System has provided for both blanking and non-blanking screen savers. Timeouts associated with these built-in screen saver mechanisms are limited to idle (dwell) time, and a change timeout that specifies the change interval for non-blanking screen savers.

The United States' Environmental Protection Agency (EPA) Energy Star program requires that monitors power down after some idle time by default. While it is possible to simply overload the existing screen saver timeouts, this solution leaves the non-privileged user little to no control over the DPMS characteristics of his or her system. For example, disabling DPMS would require some unintended side effect in the core screen saver, such as disabling the changing of a non-blanking screen saver. Providing clients with this control requires an extension to the core X Window System Protocol, and this extension seeks to fill this gap.

The design goal of the DPMS extension is to be a logical extension to the traditional screen saver. The protocol and sample implementation is designed to use the same date types and time units as the screen saver. The sample implementation works independently from the screen saver so that policy as it pertains to the interaction between screen saver and DPMS can be deferred to the user or screen saver application. The extension has been tested with and shown to work correctly with both the internal blanking and non-blanking screen savers, as well as with screen saver extension clients.

The DPMS extension is designed to be simple, yet export sufficient VESA DPMS information to enable full function clients to be written. Included is the ability to sense DPMS capability, set and get DPMS timeouts, enable and disable individual DPMS modes, enable and disable DPMS (without destroying timeout values), and sense current DPMS on/off state and power level.

There are four power levels specified by the Video Electronics Standards Association (VESA) Display Power Management Signaling (DPMS) standard. These are:

`DPMS Extension Power Levels`

0	DPMSModeOn	In use
1	DPMSModeStandby	Blanked, low power
2	DPMSModeSuspend	Blanked, lower power
3	DPMSModeOff	Shut off, awaiting activity

It is logical to assume that successive DPMS modes be chronologically at the same time or later than one another, and the protocol is designed to enforce this rule.

Note however that a concious decision is made to decouple the timeouts associated with screen saver from the DPMS timeouts. While it might be considered logical to require that the first non-zero DPMS timeout be greater than or equal to the screen saver timeout, this is intentionally omitted, leaving this policy decision to the user

---

[1] *X Window System* is a trademark of The Open Group.

or the screen saver application. In the case of a laptop where power may be scarce, the importance of power savings should supersede the screen saver. If the laptop user plugs the unit in and power is no longer a scarce commodity, it may be decided to make DPMS less aggressive, or disable it completely.

Chapter 2. Requests

`DPMSGetVersion`

client_major_version: CARD16
client_minor_version: CARD16
=>
server_major_version: CARD16
server_minor_version: CARD16

If supplied, the *client_major_version* and *client_minor_version* indicate what version of the protocol the client wants the server to implement. The server version numbers returned indicate the protocol this extension actually supports. This might not equal the version sent by the client. An implementation can (but need not) support more than one version simultaneously. The *server_major_version* and the *server_minor_version* are a mechanism to support future revisions of the Display Power Management Signaling protocol which may be necessary. In general, the major version would increment for incompatible changes, and the minor version would increment for small, upward-compatible changes. Servers that support the protocol defined in this document will return a *server_major_version* of one (1), and a *server_minor_version* of one (1).

`DPMSCapable`

=>
capable: BOOL

This request is used to determine whether or not the currently running server's devices are capable of DPMS operations. The truth value of this request is implementation defined, but is generally based on the capabilities of the graphic card and monitor combination. Also, the return value in the case of heterogeneous multi-head servers is implementation defined.

`DPMSGetTimeouts`

=>
standby_timeout: CARD16
suspend_timeout: CARD16
off_timeout: CARD16

This request returns the current values of the DPMS timeout values. All values are in units of seconds.

standby_timeout is the amount of time of inactivity before standby mode is invoked. The actual effects of this mode are implementation defined, but in the case of DPMS compliant hardware, it is implemented by shutting off the horizontal sync signal, and pulsing the vertical sync signal. Standby mode provides the quickest monitor recovery time. Note also that many monitors implement this mode identically to suspend mode. A value of zero indicates that this mode is disabled.

suspend_timeout is the amount of time of inactivity before the second level of power savings is invoked. Suspend mode's physical and electrical characteristics are implementation defined, but in DPMS compliant hardware, results in the pulsing of the horizontal sync signal, and shutting off of the vertical sync signal. Suspend mode recovery is considered to be slower than standby mode, but faster than off mode, however this is monitor dependent. As noted above, many monitors implement this mode identically to standby mode. A value of zero indicates that this mode is disabled.

off_timeout is the amount of time of inactivity before the third and final level of power savings is invoked. Off mode's physical and electrical characteristics are implementation defined, but in DPMS compliant hardware, is implemented by shutting off both horizontal and vertical sync signals, resulting in the power-down of the monitor. Recovery time is implementation dependant, but frequently is similar to the power-up time of the monitor. A value of zero indicates that this mode is disabled.

`DPMSSetTimeouts`

standby_timeout: CARD16

suspend_timeout: CARD16

off_timeout: CARD16

=>

All values are in units of seconds. *standby_timeout* is the amount of time of inactivity before standby mode will be invoked. This is the lightest level of power savings, and the monitor is generally immediately ready upon detection of user activity. This is most often implemented by shutting off the horizontal sync signal to the monitor. A value of zero disables this mode.

The *suspend_timeout* specifies the amount of time of inactivity before the screen is placed into suspend mode. Suspend mode is the middle level of power savings, resulting in a slightly longer recovery upon detection of activity. Suspend mode is most often implemented by pulsing the horizontal sync signal, and removing the vertical sync signal. A value of zero disables this mode.

The *off_timeout* specifies the amount of time of inactivity before the monitor is shut off. Off mode is the deepest level of power management, resulting in the greatest power savings and the longest recovery time. Off mode is most often implemented by removing both the horizontal and vertical signals. A value of zero disables this mode.

The values of successive power levels must be greater than or equal to the value of the previous (non-zero) level. A BadValue error is generated if an illegal combination is detected.

`DPMSEnable`

=>

This request enables the DPMS characteristics of the server, using the server's currently stored timeouts. If DPMS is already enabled, no change is effected.

`DPMSDisable`

=>

This request disables the DPMS characteristics of the server. It does not affect the core or extension screen savers. If DPMS is already disabled, no change is effected. This request is provided so that DPMS may be disabled without damaging the server's stored timeout values.

`DPMSForceLevel`

power_level: CARD16

=>

This request forces a specific DPMS level on the server. If DPMS is disabled, a BadMatch error is generated. If an erroneous power level is specified, a BadValue error is returned, and the error value contains the bad value. If the power level specified is already in effect, no changes occur. Power Level must be one of DPMSModeOn, DPMSModeStandby, DPMSModeSuspend or DPMSModeOff.

`DPMSInfo`

=>

power_level: CARD16

state: BOOL

This request returns information about the current DPMS state of the display. *state* is one of DPMSEnabled or DPMSDisabled. If *state* is DPMSEnabled, *power_level* is returned as one of DPMSModeOn, DPMSModeStandby, DPMSModeSuspend or DPMSModeOff, otherwise it is undefined.

Chapter 3. Events and Errors

No new events or errors are defined by this extension.

Chapter 4. Encoding

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

The name of this extension is "DPMS".

```
DPMSGetVersion
     1     CARD8          opcode
     1     0              DPMS opcode
     2     2              request length
     2     CARD16         client_major_version
     2     CARD16         client_minor_version
=>
     1     1              Reply
     1                    unused
     2     CARD16         sequence number
     4     0              length
     2     CARD16         server_major_version
     2     CARD16         server_minor_version
     20                   unused
```

```
DPMSCapable
     1     CARD8          opcode
     1     1              DPMS opcode
     2     1              request length
=>
     1     1              Reply
     1                    unused
     2     CARD16         sequence number
     4     0              length
     1     BOOL           capable
     23                   unused
```

```
DPMSGetTimeouts
     1     CARD8          opcode
     1     2              DPMS opcode
     2     1              request length
=>
     1     1              Reply
     1                    unused
     2     CARD16         sequence number
     4     0              length
     2     CARD16         standby_timeout
     2     CARD16         suspend_timeout
     2     CARD16         off_timeout
     18                   unused
```

```
DPMSSetTimeouts
     1     CARD8          opcode
```

```
     1     3              DPMS opcode
     2     3              request length
     2     CARD16         standby_timeout
     2     CARD16         suspend_timeout
     2     CARD16         off_timeout
     2                    unused
=>
```

DPMSEnable

```
     1     CARD8          opcode
     1     4              DPMS opcode
     2     1              request length
 =>
```

DPMSDisable

```
     1     CARD8          opcode
     1     5              DPMS opcode
     2     1              request length
 =>
```

DPMSForceLevel

```
     1     CARD8          opcode
     1     6              DPMS opcode
     2     2              request length
     2                    power_level
          0     DPMSModeOn
          1     DPMSModeStandby
          2     DPMSModeSuspend
          3     DPMSModeOff
     2                    unused
=>
```

DPMSInfo

```
     1     CARD8          opcode
     1     7              DPMS opcode
     2     1              request length
=>
     1     1              Reply
     1                    unused
     2     CARD16         sequence number
     4     0              length
     2                    power_level
           0     DPMSModeOn
           1     DPMSModeStandby
           2     DPMSModeSuspend
           3     DPMSModeOff
     1     BOOL           state
     21                   unused
```