

# Defined Types

---

## DPSTextContextRec

**DECLARED IN** dpsclient/dpsfriends.h

**SYNOPSIS** typedef struct \_t\_DPSTextContextRec {  
    char \***priv**;  
    DPSSpace **space**;  
    DPSTextProgramEncoding **programEncoding**;  
    DPSTextNameEncoding **nameEncoding**;  
    struct \_t\_DPSTextProcsRec const \* **procs**;  
    void (\***textProc**)();  
    void (\***errorProc**)();  
    DPSTextResults **resultTable**;  
    unsigned int **resultTableLength**;  
    struct \_t\_DPSTextContextRec \***chainParent**, \***chainChild**;  
    DPSTextContextType **type**;  
} **DPSTextContextRec**, \***DPSTextContext**;

**DESCRIPTION** The **DPSTextContextRec** structure represents a Display PostScript context.

---

## DPSTextContextType

**DECLARED IN** dpsclient/dpsfriends.h

**SYNOPSIS** typedef enum {  
    **dps\_machServer**,  
    **dps\_fdServer**,  
    **dps\_stream**  
} **DPSTextContextType**;

**DESCRIPTION** These represent the context types supported by NeXT's version of Display PostScript, as used in the **type** field of a **DPSTextContextRec** structure.

---

## DPSErrorCode

**DECLARED IN** dpsclient/dpsclient.h

**SYNOPSIS** typedef enum \_DPSErrorCode {  
    **dps\_err\_ps** = DPS\_ERROR\_BASE,  
    **dps\_err\_nameTooLong**,  
    **dps\_err\_resultTagCheck**,  
    **dps\_err\_resultTypeCheck**,  
    **dps\_err\_invalidContext**,  
    **dps\_err\_select** = DPS\_NEXT\_ERROR\_BASE,  
    **dps\_err\_connectionClosed**,  
    **dps\_err\_read**,  
    **dps\_err\_write**,  
    **dps\_err\_invalidFD**,  
    **dps\_err\_invalidTE**,  
    **dps\_err\_invalidPort**,  
    **dps\_err\_outOfMemory**,  
    **dps\_err\_cantConnect**  
} **DPSErrorCode**;

**DESCRIPTION** Error codes passed to a **DPSErrorProc()** function.

---

## DPSEventFilterFunc

**DECLARED IN** dpsclient/dpsNeXT.h

**SYNOPSIS** typedef int (\***DPSEventFilterFunc**)(NXEvent \**ev* );

**DESCRIPTION** Call-back function used to filter events.

---

## DPSFDProc

**DECLARED IN** dpsclient/dpsNeXT.h

**SYNOPSIS** typedef void (\***DPSFDProc**)( int *fd*, void \**userData* );

**DESCRIPTION** Call-back function used when a file descriptor is registered through **DPSAddFD()**.

---

## DPSNumberFormat

**DECLARED IN** dpsclient/dpsNeXT.h

**SYNOPSIS** typedef enum \_DPSNumberFormat {  
#ifdef \_\_BIG\_ENDIAN\_\_  
    **dps\_float** = 48,  
    **dps\_long** = 0,  
    **dps\_short** = 32  
#else  
    **dps\_float** = 48+128,  
    **dps\_long** = 0+128,  
    **dps\_short** = 32+128  
} **DPSNumberFormat**;

**DESCRIPTION** These constants are used by the **DPSDoUserPath()** function to describe the type of numbers that are being passed.

---

## DPSPingProc

**DECLARED IN** dpsclient/dpsNeXT.h

**SYNOPSIS** typedef void (\***DPSPingProc**)  
    (DPSContext *ctxt*,  
    void \**userData*);

**DESCRIPTION** Call-back function used by **DPSAsynchronousWaitContext()**.

---

### **DPSPortProc**

**DECLARED IN** dpsclient/dpsNeXT.h

**SYNOPSIS** typedef void (\***DPSPortProc**)  
( msg\_header\_t \*msg,  
void \*userData );

**DESCRIPTION** Call-back function used when a port is registered through **DPSAddPort()**.

---

### **DPSTimedEntry**

**DECLARED IN** dpsclient/dpsNeXT.h

**SYNOPSIS** typedef struct \_\_DPSTimedEntry \***DPSTimedEntry**;

**DESCRIPTION** The return type for **DPSAddTimedEntry()**.

---

### **DPSTimedEntryProc**

**DECLARED IN** dpsclient/dpsNeXT.h

**SYNOPSIS** typedef void (\***DPSTimedEntryProc**)  
(DPSTimedEntry *timedEntry*,  
double *now*,  
void \*userData );

**DESCRIPTION** Call-back function used when a timed entry is registered through **DPSAddTimedEntry()**.

---

## DPSUserPathAction

**DECLARED IN** dpsclient/dpsNeXT.h

**SYNOPSIS** typedef enum \_DPSUserPathAction {  
    **dps\_uappend,**  
    **dps\_ufill,**  
    **dps\_ueofill,**  
    **dps\_ustroke,**  
    **dps\_ustrokepath,**  
    **dps\_inufill,**  
    **dps\_inueofill,**  
    **dps\_inustroke,**  
    **dps\_def,**  
    **dps\_put**  
} **DPSUserPathAction;**

**DESCRIPTION** These constants are convenient representations of some of the PostScript operator indices, suitable for enrollment in the action array passed to **DPSDoUserPath()**.

---

## DPSUserPathOp

**DECLARED IN** dpsclient/dpsNeXT.h

**SYNOPSIS** typedef enum \_DPSUserPathOp {  
    **dps\_setbbox,**  
    **dps\_moveto,**  
    **dps\_rmoveto,**  
    **dps\_lineto,**  
    **dps\_rlineto,**  
    **dps\_curveto,**  
    **dps\_rcurveto,**  
    **dps\_arc,**  
    **dps\_arn,**  
    **dps\_arct,**  
    **dps\_closepath,**  
    **dps\_ucache**  
} **DPSUserPathOp**;

**DESCRIPTION** These constants represent the PostScript operators that can be passed in **DPSDoUserPath()**'s operator array.

---

## NXCoord

**DECLARED IN** dpsclient/event.h

**SYNOPSIS** typedef float **NXCoord**

**DESCRIPTION** Used to represent a single coordinate in a Cartesian coordinate system.

---

## NXEvent

**DECLARED IN**   dpsclient/event.h

**SYNOPSIS**   typedef struct \_NXEvent {  
                  int **type**;  
                  NXPoint **location**;  
                  long **time**;  
                  int **flags**;  
                  unsigned int **window**;  
                  NXEventData **data**;  
                  DPSText **ctxt**;  
          } **NXEvent**, \***NXEventPtr**;

**DESCRIPTION**   Represents a single event; this structure is also known as the *event record*. The fields are:

type	The type of event (see “Event Types,” below)
location	The event’s location in the base coordinate system of its window
time	The time of the event (in hardware-dependent units) since system startup
flags	Mouse-button and modifier-key flags (see “Event Flags,” below)
window	The window number of the window associated with the event
data	Additional type-specific data (see “NXEventData,” below)
ctxt	The PostScript context of the event

---

## NXEventData

**DECLARED IN**    dpsclient/event.h

**SYNOPSIS**    typedef union {  
                struct {  
                    short **eventNum**;  
                    int **click**;  
                    unsigned char **pressure**;  
                } **mouse**;  
                struct {  
                    short **repeat**;  
                    unsigned short **charSet**;  
                    unsigned short **charCode**;  
                    unsigned short **keyCode**;  
                    short **keyData**;  
                } **key**;  
                struct {  
                    short **eventNum**;  
                    int **trackingNum**;  
                    int **userData**;  
                } **tracking**;  
                struct {  
                    short **subtype**;  
                    union {  
                        float **F**[2];  
                        long **L**[2];  
                        short **S**[4];  
                        char **C**[8];  
                    } **misc**;  
                } **compound**;  
    } **NXEventData**;

**DESCRIPTION**    This structure supplies type-specific information for an event. It's a union of four structures, where the type of the event determines which structure is pertinent:

- **mouse** is used for mouse events.
- **key** is used for keyboard events.
- **tracking** is for tracking-rectangle events.
- **compound** is for system-, kit-, and application-defined events.



---

## **NXPoint**

<b>DECLARED IN</b>	dpsclient/event.h
<b>SYNOPSIS</b>	<pre>typedef struct _NXPoint {     NXCoord <b>x</b>;     NXCoord <b>y</b>; } <b>NXPoint</b>;</pre>
<b>DESCRIPTION</b>	Represents a point in a Cartesian coordinate system.

---

## **NXSize**

<b>DECLARED IN</b>	dpsclient/event.h
<b>SYNOPSIS</b>	<pre>typedef struct _NXSize {     NXCoord <b>width</b>;     NXCoord <b>height</b>; } <b>NXSize</b>;</pre>
<b>DESCRIPTION</b>	Represents a two-dimensional size.

# Symbolic Constants

---

## All Contexts

DECLARED IN	dpsclient/dpsNeXT.h
SYNOPSIS	DPS_ALLCONTEXTS
DESCRIPTION	This constant represents all extant contexts.

---

## Alpha Constants

DECLARED IN	dpsclient/dpsNeXT.h
SYNOPSIS	NX_DATA NX_ONES
DESCRIPTION	These constants represent alpha values.

---

## Character Set Values

DECLARED IN	dpsclient/event.h
SYNOPSIS	NX_ASCIISET NX_SYMBOLSET NX_DINGBATSSET
DESCRIPTION	These constants represent the values that may occur in the <b>data.key.charSet</b> field of an NXEvent structure.

---

## Compositing Operations

**DECLARED IN** dpsclient/dpsNeXT.h

**SYNOPSIS** NX\_CLEAR  
NX\_COPY  
NX\_SOVER  
NX\_SIN  
NX\_SOUT  
NX\_SATOP  
NX\_DOVER  
NX\_DIN  
NX\_DOUT  
NX\_DATOP  
NX\_XOR  
NX\_PLUSD  
NX\_HIGHLIGHT  
NX\_PLUSL

**DESCRIPTION** These represent the compositing operations used by **PScomposite()** and the NXImage class.

---

## Error Code Bases

**DECLARED IN** dpsclient/dpsclient.h

**SYNOPSIS** DPS\_ERROR\_BASE  
DPS\_NEXT\_ERROR\_BASE

**DESCRIPTION** These constants represent the lowest values for Display PostScript error codes.

---

## Event Types

**DECLARED IN**    dpsclient/event.h

<b>SYNOPSIS</b>	<b>Type</b>	<b>Meaning</b>
	NX_NULLEVENT	A non-event
	NX_LMOUSEDOWN	Left mouse-down
	NX_LMOUSEUP	Left mouse-up
	NX_LMOUSEDRAGGED	left mouse-dragged
	NX_MOUSEDOWN	Same as NX_LMOUSEDOWN
	NX_MOUSEUP	Same as NX_LMOUSEUP
	NX_MOUSEDRAGGED	Same as NX_LMOUSEDRAGGED
	NX_RMOUSEDOWN	Right mouse-down
	NX_RMOUSEUP	Right mouse-up
	NX_RMOUSEDRAGGED	Right mouse-dragged
	NX_MOUSEMOVED	Mouse-moved
	NX_MOUSEENTERED	Mouse-entered
	NX_MOUSEEXITED	Mouse-exited
	NX_KEYDOWN	Key-down
	NX_KEYUP	Key-up event
	NX_FLAGSCHANGED	Flags-changed
	NX_KITDEFINED	Application Kit-defined
	NX_SYSDEFINED	System-defined
	NX_APPDEFINED	Application-defined
	NX_TIMER	Timer used for tracking
	NX_CURSORUPDATE	Cursor tracking
	NX_JOURNALEVENT	Event used by journaling
	NX_FIRSTEVENT	The smallest-valued event constant
	NX_LASTEVENT	The greatest-valued event constant
	NX_ALLEVENTS	A value that includes all event types

**DESCRIPTION**    These constants represent event types. They're passed as the **type** field of the NXEvent structure that's created when an event occurs.

---

## Event Type Masks

**DECLARED IN**    dpsclient/event.h

**SYNOPSIS**    NX\_NULLEVENTMASK  
              NX\_LMOUSEDOWNMASK  
              NX\_LMOUSEUPMASK  
              NX\_RMOUSEDOWNMASK  
              NX\_RMOUSEUPMASK  
              NX\_MOUSEMOVEDMASK  
              NX\_LMOUSEDRAGGEDMASK  
              NX\_RMOUSEDRAGGEDMASK  
              NX\_MOUSEENTEREDMASK  
              NX\_MOUSEEXITEDMASK  
              NX\_KEYDOWNMASK  
              NX\_KEYUPMASK  
              NX\_FLAGSCHANGEDMASK  
              NX\_KITDEFINEDMASK  
              NX\_APPDEFINEDMASK  
              NX\_SYSDEFINEDMASK  
              NX\_TIMERMASK  
              NX\_CURSORUPDATESMASK  
              NX\_MOUSEDOWNMASK  
              NX\_MOUSEUPMASK  
              NX\_MOUSEDRAGGEDMASK  
              NX\_JOURNALEVENTMASK

**DESCRIPTION**    These masks correspond to the event types defined immediately above. They let you query the **type** field of an NXEvent structure for the existence of a particular event type.

---

## Forever

**DECLARED IN**    dpsclient/dpsNeXT.h

**SYNOPSIS**    NX\_FOREVER

**DESCRIPTION**    A long, long time. Typically used as the timeout argument to **DPSGetEvent()**.

---

## Keyboard State Flags Masks

**DECLARED IN**    dpsclient/event.h

<b>SYNOPSIS</b>	<b>Type</b>	<b>Meaning</b>
	NX_ALPHASHIFTMASK	Shift lock
	NX_SHIFTMASK	Shift key
	NX_CONTROLMASK	Control key
	NX_ALTERNATEMASK	Alt key
	NX_COMMANDMASK	Command key
	NX_NUMERICPADMASK	Number pad key
	NX_HELPMASK	Help key
	NX_NEXTCTRLKEYMASK	Control key
	NX_NEXTLSHIFTKEYMASK	Left shift key
	NX_NEXTRSHIFTKEYMASK	Right shift key
	NX_NEXTLCMDKEYMASK	Left command key
	NX_NEXTRCMDKEYMASK	Right command key
	NX_NEXTLALTKEYMASK	Left alt key
	NX_NEXTRALTKEYMASK	Right alt key

**DESCRIPTION**    These masks correspond to keyboard states that might be included in an NXEvent structure's **flags** mask. The masks are grouped as device-independent (NX\_ALPHASHIFTMASK through NX\_HELPMASK) and device-dependent (all others).

---

## Miscellaneous Event Flags Masks

**DECLARED IN**    dpsclient/event.h

<b>SYNOPSIS</b>	<b>Type</b>	<b>Meaning</b>
	NX_STYLUSPROXIMITYMASK	Stylus is in proximity (for tablets)
	NX_NONCOALSESCEDMASKE	Event coalescing disabled

**DESCRIPTION**    These masks correspond to miscellaneous states that might be included in an NXEvent structure's **flags** mask.

---

## Window Backing Types

<b>DECLARED IN</b>	dpsclient/dpsNeXT.h
<b>SYNOPSIS</b>	NX_RETAINED NX_NONRETAINED NX_BUFFERED
<b>DESCRIPTION</b>	These represent the three backing types provided by window devices (and used by the Application Kit's Window objects).

---

## Window Screen List Placement

<b>DECLARED IN</b>	dpsclient/dpsNeXT.h
<b>SYNOPSIS</b>	NX_ABOVE NX_BELOW NX_OUT
<b>DESCRIPTION</b>	These represent the placement of a window device in the screen list.

