

# 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**,

<b>DESCRIPTION</b>	Error codes passed to a <b>DPSErrorProc()</b> function.
--------------------	---

**DECLARED IN**      `dpsclient/dpsNeXT.h`

<b>DESCRIPTION</b>	Call-back function used to filter events.
--------------------	---

**DECLARED IN**      `dpsclient/dpsNeXT.h`

<b>DESCRIPTION</b>	Call-back function used when a file descriptor is registered through <b>DPSAddFD()</b> .
--------------------	--

**DECLARED IN**      `dpsclient/dpsNeXT.h`

**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);`

<b>DESCRIPTION</b>	Call-back function used by <b>DPSAsynchronousWaitContext()</b> .
--------------------	--

## DPSPortProc

**DECLARED IN**      `dpsclient/dpsNeXT.h`

**SYNOPSIS** `typedef void (*DPSPortProc)`  
`( msg_header_t *msg,`  
`void *userData );`

<b>DESCRIPTION</b>	Call-back function used when a port is registered through <b>DPSAddPort()</b> .
--------------------	---

## DPSTimedEntry

**DECLARED IN**      `dpsclient/dpsNeXT.h`

```

SYNOPSIS
*DPSTimedEntry;
typedef struct __DPSTimedEntry

```

<b>DESCRIPTION</b>	The return type for <b>DPSAddTimedEntry()</b> .
--------------------	---

## DPSTimedEntryProc

**DECLARED IN**      `dpsclient/dpsNeXT.h`

```

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

```

<b>DESCRIPTION</b>	Call-back function used when a timed entry is registered through <b>DPSAddTimedEntry()</b> .
--------------------	--

## DPSUserPathAction

**DECLARED IN**      `dpsclient/dpsNeXT.h`

```

SYNOPSIS
{
    dps_uappend,
    dps_ufill,
}
typedef enum _DPSUserPathAction

```

```

    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
    dps_setbbox,
    dps_moveto,
    dps_rmoveto,
    dps_lineto,
    dps_rlineto,
    dps_curveto,
    dps_rcurveto,
    dps_arc,
    dps_arcn,
    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**

<b>DESCRIPTION</b>	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;  
    DPSCContext ctxt;  
} NXEvent, *NXEventPtr;
```

<b>DESCRIPTION</b>	Represents a single event; this structure is also known as the <i>event record</i> . 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.



DECLARED IN

dpsclient/event.h

SYNOPSIS

NX\_ASCIISET  
NX\_SYMBOLSET  
NX\_DINGBATSSET

DESCRIPTION

These constants represent the values that may occur in the **data.key.charSet** 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	
SYNOPSIS		
Meaning		Type
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	





Forever

DECLARED INdpsclient/dpsNeXT.h

SYNOPSISNX\_FOREVER

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

Keyboard State Flags Masks

DECLARED INdpsclient/event.h

Meaning	SYNOPSIS	Type
	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

DESCRIPTIONThese 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 INdpsclient/event.h

Meaning	SYNOPSIS	Type
	NX_STYLUSPROXIMITYMASK	Stylus is in proximity (for tablets)
	NX_NONCOALSESCEDMASK	Event coalescing disabled

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

Window Backing Types

DECLARED INdpsclient/dpsNeXT.h

SYNOPSIS

NX\_NONRETAINED  
NX\_BUFFERED

NX\_RETAINED

DESCRIPTION

These represent the three backing types provided by window devices (and used by the Application Kit's Window objects).

Window Screen List Placement

DECLARED IN

dpsclient/dpsNeXT.h

SYNOPSIS

NX\_BELOW  
NX\_OUT

NX\_ABOVE

DESCRIPTION

These represent the placement of a window device in the screen list.