Client Library Functions

Controlling a PostScript Execution Context

Create a context

DPSContext **DPSCreateContext(**const char ***hostName*, const char ***serverName*, DPSTextProc *textProc*, DPSErrorProc *errorProc***)**[2]

DPSContext **DPSCreateContextWithTimeoutFromZone(**const char ***hostName*, const char ***serverName*, DPSTextProc *textProc*, DPSErrorProc *errorProc*, int *timeout*, NXZone ***zone***)**[2]

DPSContext **DPSCreateNonsecureContext(**const char ***hostName*, const char ***serverName*, DPSTextProc *textProc*, DPSErrorProc *errorProc*, int *timeout*, NXZone ***zone***)**[2]

DPSContext **DPSCreateStreamContext(**NXStream ***stream*, int *debugging*, DPSProgramEncoding *progEnc*, DPSNameEncoding *nameEnc*, DPSErrorProc *errorProc***)**[2]

void **DPSDestroyContext(**DPSContext *context***)**

Create a child context

int **DPSChainContext(**DPSContext *parent*, DPSContext *child***)**

void **DPSUnchainContext(**DPSContext *context***)**

Access the current context

void **DPSSetContext(**DPSContext *context***)**

DPSContext **DPSGetCurrentContext(**void**)**

Control a context

int **DPSSynchronizeContext(**DPSContext *context*, int *enableFlag***)**[2]

void **DPSWaitContext(**DPSContext *context***)**

void **DPSAsynchronousWaitContext(**DPSContext *context*, DPSPingProc *handler*, void ***userData***)**

Warning: The following two context-controlling functions aren't implemented in NeXTSTEP

void **DPSInterruptContext()**

void **DPSResetContext()**

Extract space from a context

DPSSpace **DPSSpaceFromContext(**DPSContext *context***)**

Destroy a space and all contexts in it

void **DPSDestroySpace(**DPSSpace *space***)**

Sending Data to the Window Server

Send PostScript code to the Window Server

void **DPSWritePostScript(**DPSContext *context*, const void **buf*, int *count***)**
void **DPSWriteData(**DPSContext *context*, const void **buf*, unsigned int *count***)**
void **DPSPrintf(**DPSContext *context*, const char **format*, ...**)**
void **DPSFlushContext(**DPSContext *context***)**
void **DPSFlush(**void**)**[2]
void **DPSSendEOF(**DPSContext *context***)**[2]

Send an encoded PostScript path to the Window Server

void **DPSDoUserPath(**void **coords*, int *numCoords*, DPSNumberFormat *numType*, unsigned char **ops*, int *numOps*, void **bbox*, int *action***)**[2]
void **DPSDoUserPathWithMatrix(**void **coords*, int *numCoords*, DPSNumberFormat *numType*, unsigned char **ops*, int *numOps*, void **bbox*, int *action*, float *matrix[6]***)**[2]

User Objects and User Names

Create a user object

int **DPSDefineUserObject(**int *index***)**[2]
void **DPSUndefineUserObject(**int *index***)**[2]

Access the system and user name tables

void **DPSMapNames(**DPSContext *context*, unsigned int *numNames*, const char *const **nameArray*, long int *const **numPtrArray***)**
const char * **DPSNameFromIndex(**int *index***)**
const char * **DPSNameFromTypeAndIndex(**short *type*, int *index***)**[2]

Event-Handling

Access events from the Window Server

int **DPSGetEvent(**DPSContext *context*, NXEvent **anEvent*, int *mask*, double *timeout*, int *threshold***)**[2]
int **DPSPeekEvent(**DPSContext *context*, NXEvent **anEvent*, int *mask*, double *timeout*, int *threshold***)**[2]
void **DPSDiscardEvents(**DPSContext *context*, int *mask***)**[2]

Coalesce events

int **DPSSetTracking(**int *flag***)**[2]

Set the event-filter function

DPSEventFilterFunc **DPSSetEventFunc(**DPSContext *context*, DPSEventFilterFunc *func***)**[2]

Create an event

int **DPSPostEvent(**NXEvent ***anEvent*, int *atStart***)**[2]

Create a timed entry

DPSTimedEntry **DPSAddTimedEntry(**double *period*, DPSTimedEntryProc *handler*, void ***userData*, int *priority***)**[2]

void **DPSRemoveTimedEntry(**DPSTimedEntry *teNumber***)**[2]

Initiate a count down for the wait cursor

void **DPSStartWaitCursorTimer(**void**)**[2]

Allow dead key processing

void **DPSSetDeadKeysEnabled(**DPSContext *context*, int *flag***)**[2]

Generate an event mask for an event type

int **NX_EVENTCODEMASK(**int *type***)**

File and Port Monitoring

Monitor a file descriptor

void **DPSAddFD(**int *fd*, DPSFDProc *handler*, void ***userData*, int *priority***)**[2]

void **DPSRemoveFD(**int *fd***)**[2]

Monitor a Mach port

void **DPSAddPort(**port_t *newPort*, DPSPortProc *handler*, int *maxSize*, void ***userData*, int *priority***)**[2]

void **DPSRemovePort(**port_t *port***)**[2]

Set the notify port call-back function

void **DPSAddNotifyPortProc(**DPSPortProc *handler*, void ***userData***)**[2]

void **DPSRemoveNotifyPortProc(**DPSPortProc *handler***)**[2]

Text-Handling

Set the text call-back functions

DPSTextProc **DPSSetTextProc(**DPSContext *context*, DPSTextProc *tp***)**
DPSTextProc **DPSSetTextBackstop(**DPSTextProc *textProc***)**
DPSTextProc **DPSGetCurrentTextBackstop(***void***)**

Debugging and Error-Handling

Trace data and events

int **DPSTraceContext(**DPSContext *context*, int *flag***)**[2]
void **DPSTraceEvents(**DPSContext *context*, int *flag***)**[2]

Handle errors

DPSErrorProc **DPSSetErrorProc(**DPSContext *context*, DPSErrorProc *ep***)**
void **DPSDefaultErrorProc(**DPSContext *context*, DPSErrorCode *errorCode*, long unsigned int arg_1, long unsigned int arg_2**)**
void **DPSSetErrorBackstop(**DPSErrorProc *errorProc***)**
DPSErrorProc **DPSGetCurrentErrorBackstop(***void***)**
void **DPSPrintError(**FILE **fp*, const DPSBinObjSeqRec *error***)**[2]
void **DPSPrintErrorToStream(**NXStream **stream*, const DPSBinObjSeqRec *error***)**[2]

Functions Used by pswrap

Wait for return values from the Window Server

void **DPSAwaitReturnValues(**DPSContext *context***)**

Write strings in binary object sequence

void **DPSWriteStringChars(**DPSContext *context*, const char **buf*, unsigned int *count***)**

Write PostScript code in a specified format

void **DPSWriteTypedObjectArray(**DPSContext *context*, DPSDefinedType *type*, const void **array*, unsigned int *length***)**

Begin a new binary object sequence

void **DPSBinObjSeqWrite(**DPSContext *context*, const void **buf*, unsigned int *count***)**

Define information expected from the PostScript interpreter

void **DPSSetResultTable(**DPSContext *context*, DPSResults *table*, unsigned int *length***)**

Update a context's name map from the client library's name map

void **DPSUpdateNameMap(**DPSContext *context***)**