Print Service Group

Print Service Group

Print Overview

Print: Guides

Distributed Print

Print Service Group

Distributed Print: Functions

NWDPASAddAttribute

Adds an attribute to an attribute set
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_atr.h>

N_EXTERN_LIBRARY (nint) NWDPASAddAttribute (
   NWDPAccessorRef   accessorRef,
   NWDPASAVPRef      avpRef,
   pNWDPOid          attributeIdPtr,
   pnuint32          qualifierPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

avpRef

(IN) Points to the NWDPASAVPStruct structure identifying the attribute set where the attribute is added.

attributeIdPtr

(IN) Points to the NWDPOid structure containing the OID (Object Identifier) representing the attribute to be added (see the nwdp_wko.ogh file for a list of OIDs).

qualifierPtr

(IN) Points to the attribute qualifier (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

FF	

Remarks

If the attribute is already in the attribute set, its values are removed. The attribute pointer of the *avpRef* parameter points to the new attribute. The value pointer of the *avpRef* parameter is set to NULL because no values exist.

If the attribute set is going to be used for MODB (Managed Object Database) or NDS modifications, you can set the modify operator for each attribute before performing the modify function. Pass a pointer to an enumeration from the NWDPASModifyOperatorEnum for the *qualifierPtr* parameter.

If the *avpRef* parameter was not created when the attribute set was created, call the **NWDPASMakeAVPRef** function.

If the **NWDPASAddAttribute** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* parameter contains the following:

0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPASRemoveAttribute

NWDPASAddAttrValue

Adds a value to the specified attribute
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_atr.h>

N_EXTERN_LIBRARY (nint) NWDPASAddAttrValue (
   NWDPAccessorRef       accessorRef,
   NWDPASAVPRef          avpRef,
   pNWDPOid              attributeIdPtr,
   pNWDPAttributeValue   newValuePtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

avpRef

(IN) Points to the NWDPASAVPStruct structure identifying the attribute set where the attribute value is added.

attributeIdPtr

(IN) Points to the NWDPOid structure containing the OID (Object Identifier) representing the attribute to which the value is added (see the nwdp_wko.ogh file for a list of OIDs).

newValuePtr

(IN) Points to the NWDPAttributeValue structure containing the new value.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFF	N_FAILURE

0xFFFFFFFF	N_FAILURE

Remarks

If the attribute has existing values, the new value is appended to the end of the list. The value pointer of the *avpRef* parameter is updated to point to the new value.

If the attribute pointer of the *avpRef* parameter already points to the correct attribute, pass NULL for the *attributeIdPtr* parameter.

If the *avpRef* parameter was not created when the attribute set was created, call the **NWDPASMakeAVPRef** function.

If the **NWDPASAddAttrValue** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPASModifyAttrValue
NWDPASRemoveAttrValue

NWDPASAttributeListCallback

Lists the attributes
Local Servers: N/A
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_atr.h>

N_TYPEDEF_CALLBACK (nint, NWDPASAttributeListCallback) (
   NWDPAccessorRef            accessorRef,
   nparam                     callerDefinedParam,
   nint                       totalCallsToBeMade,
   nint                       currentCallCount,
   pNWDPASAttributeListItem   attributeListItemPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

callerDefinedParam

(IN) Specifies a caller-defined contextual variable (optional).

totalCallsToBeMade

(IN) Specifies the number of times the **NWDPASListAttributes** function calls the **NWDPASAttributeListCallback** function (number of attributes plus one).

currentCallCount

(IN) Specifies the current call number.

attributeListItemPtr

(IN) Points to the NWDPASAttributeListItem structure.

Return Values

0x00000000	N_SUCCESS
0xFFFFFF	N_FAILURE

FF	

Remarks

For the *callerDefinedParam* parameter, this function uses the contextual variable to create the following:

a linked list head into which items can be added

a window handle where items can be displayed

any other data or combination of these two steps needed by the callback routine (optional)

For the *totalCallsToBeMade* parameter, the *attributeListItemPtr* parameter is NULL for the last callback.

The *avpRef* field returned in the NWDPASAttributeListItem structure is the same *avpRef* field passed to the **NWDPASListAttributes** function. The attribute pointer of the *avpRef* field is set to the currently listed attribute. The value pointer of the *avpRef* field is set to the first value of this attribute.

NCP Calls

None

See Also

NWDPASListAttributes
NWDPASListAttrValues

NWDPASAttrValueListCallback

Lists the attribute values
Local Servers: N/A
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_atr.h>

N_TYPEDEF_CALLBACK (nint, NWDPASAttrValueListCallback) (
   NWDPAccessorRef   accessorRef,
   nparam            callerDefinedParam,
   nint              totalCallsToBeMade,
   nint              currentCallCount,
   NWDPASAVPRef      avpRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

callerDefinedParam

(IN) Specifies a caller-defined contextual variable (optional).

totalCallsToBeMade

(IN) Specifies the number of times the **NWDPASListAttrValues** function calls the **NWDPASAttrValueListCallback** function (number of attributes plus one).

currentCallCount

(IN) Specifies the current call number.

avpRef

(IN) Points to the NWDPASAVPStruct structure.

Return Values

0x00000000	N_SUCCESS
0xFFFFFF	N_FAILURE

FF	

Remarks

For the *callerDefinedParam* parameter, this function uses the contextual variable to create the following:

a linked list head into which items can be added

a window handle where items can be displayed

any other data or combination of these two steps needed by the callback routine (optional)

For the *totalCallsToBeMade* parameter, the *avpRef* parameter is NULL for the last callback.

For the *avpRef* parameter, this is the same avpRef that was passed into the **NWDPASListAttrValues** function. The value pointer of this parameter is set to the currently listed value. You can use the value pointer to list the value.

NCP Calls

None

See Also

NWDPASListAttributes
NWDPASListAttrValues

NWDPASCreateRef

Creates an attribute set reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_atr.h>

N_EXTERN_LIBRARY (nint) NWDPASCreateRef (
   NWDPAccessorRef   accessorRef,
   nuint             sizeOfAttrSet,
   pNWDPAttrSetRef   attrSetRefPtr,
   pNWDPASAVPRef     avpRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

sizeOfAttrSet

(IN) Specifies the number of attributes for which memory needs to be allocated.

attrSetRefPtr

(OUT) Points to the NWDPAttrSetRef, the resulting attribute set reference.

avpRefPtr

(OUT) Points to the NWDPASAVPRef, the resulting attribute and value pointer reference (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

FF	

Remarks

If the attribute set grows beyond the number specified by the *sizeOfAttrSet* parameter, the library allocates more memory. If unsure of the number allowed, pass NWDP_AS_DEFAULT_SET_SIZE.

The *avpRefPtr* parameter is needed to call most attribute set functions.

If the **NWDPASCreateRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPASReleaseRef
NWDPASUseRef

NWDPASCreateRefBasedOnSet

Creates an attribute set reference from an existing attribute set
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_atr.h>

N_EXTERN_LIBRARY (nint) NWDPASCreateRefBasedOnSet (
   NWDPAccessorRef     accessorRef,
   pNWDPAttributeSet   attributeSetPtr,
   pNWDPAttrSetRef     attrSetRefPtr,
   pNWDPASAVPRef       avpRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

attributeSetPtr

(IN) Points to the NWDPAttributeSet structure containing the attribute set used to create the new attribute set reference.

attrSetRefPtr

(OUT) Points to the NWDPAttrSetRef, the resulting attribute set reference.

avpRefPtr

(OUT) Points to the NWDPASAVPRef, the resulting attribute and value pointer reference (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

FF	

Remarks

The attribute set of the new attribute set reference is a copy of the set identified by the *attributeSetPtr* parameter

The *avpRefPtr* parameter is needed to call most attribute set functions.

If the **NWDPASCreateRefBasedOnSet** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPASReleaseRef

NWDPASListAttributes

Lists the attributes identified by an attribute set reference
Local Servers: nonblocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_atr.h>

N_EXTERN_LIBRARY (nint) NWDPASListAttributes (
   NWDPAccessorRef               accessorRef,
   NWDPASAVPRef                  avpRef,
   NWDPASAttributeListCallback   listCallback,
   nparam                         callerDefinedParam);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

avpRef

(IN) Points to the NWDPASAVPStruct structure identifying the attribute set from which the attributes are listed.

listCallback

(IN) Specifies the application-supplied callback function called by the **NWDPASListAttributes** function for each attribute.

callerDefinedParam

(IN) Specifies a caller-defined parameter, containing application-specific data, that is passed to the callback function (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFF	N_FAILURE

0xFFFFFFFF	N_FAILURE

Remarks

The **NWDPASListAttributes** function is generally nonblocking, except when the associated callback function is blocking, then the **NWDPASListAttributes** function is also blocking.

If the *avpRef* parameter was not created at the same time the attribute set was created, call the **NWDPASMakeAVPRef** function.

If the **NWDPASListAttributes** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x000A0005L NWDP_LE_UNKNOWN_CALLBACK_ERR

NCP Calls

None

See Also

NWDPASListAttrValues

NWDPASListAttrValues

Lists the values of a given attribute in an attribute set reference
Local Servers: nonblocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_atr.h>

N_EXTERN_LIBRARY (nint) NWDPASListAttrValues (
   NWDPAccessorRef               accessorRef,
   NWDPASAVPRef                  avpRef,
   pNWDPOid                      attributeIdPtr,
   NWDPASAttrValueListCallback   listCallback,
   nparam                         callerDefinedParam);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

avpRef

(IN) Points to the NWDPASAVPStruct structure identifying the attribute set and attribute from which to list values.

attributeIdPtr

(IN) Points to the NWDPOid structure containing the OID identifying the attribute to be added (see the nwdp_wko.ogh file for a list of OIDs).

listCallback

(IN) Specifies the application-supplied callback function that is called by the **NWDPASListAttrValues** function to obtain each value.

callerDefinedParam

(IN) Specifies a caller-defined parameter, containing application-specific data, that is passed to the callback function (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The **NWDPASListAttrValues** function is generally nonblocking, except when the associated callback function is blocking, then the **NWDPASListAttrValues** function is also blocking.

If the attribute pointer of the *avpRef* parameter points to the correct attribute, pass NULL for the *attributeIdPtr* parameter.

If the **NWDPASListAttrValues** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x000A0005L NWDP_LE_UNKNOWN_CALLBACK_ERR

NCP Calls

None

See Also

NWDPASListAttributes

NWDPASMakeAVPRef

Assigns an attribute set to a NWDPASAVPRef (attribute and value pointer reference)

Local Servers: nonblocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_atr.h>

N_EXTERN_LIBRARY (nint) NWDPASMakeAVPRef (
   NWDPAccessorRef   accessorRef,
   NWDPAttrSetRef    attrSetRef,
   NWDPASAVPRef      avpRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

attrSetRef

(IN) Points to the NWDPAttrSetStruct structure containing the attribute set.

avpRef

(OUT) Points to the NWDPASAVPStruct structure (the attribute set reference of the NWDPASAVPStruct structure is changed to point to the *attrSetRef* parameter).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The attribute and value pointers of the *avpRef* parameter are set to the first attribute and value of the attribute set. No memory is allocated by the library during this call.

NCP Calls

None

NWDPASModifyAttrValue

Modifies an attribute value
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_atr.h>

N_EXTERN_LIBRARY (nint) NWDPASModifyAttrValue (
   NWDPAccessorRef       accessorRef,
   NWDPASAVPRef          avpRef,
   pNWDPOid              attributeIdPtr,
   pNWDPAttributeValue   oldValuePtr,
   pNWDPAttributeValue   newValuePtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

avpRef

(IN) Points to the NWDPASAVPStruct structure identifying the attribute set where the value is modified.

attributeIdPtr

(IN) Points to the NWDPOid structure containing the OID identifying the attribute whose values are modified (see the nwdp_wko.ogh file for a list of OIDs).

oldValuePtr

(IN) Points to the NWDPAttributeValue structure containing the value being modified.

newValuePtr

(IN) Points to the NWDPAttributeValue structure containing the new value.

Return Values

0x0000000	N_SUCCESS

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the attribute pointer of the *avpRef* parameter already points to the correct attribute, pass NULL for the*attributeIdPtr* parameter.

If the value pointer of the *avpRef* parameter already points to the correct value, or if no value exists, pass NULL for the*oldValuePtr* parameter. The value identified by the value pointer of the *avpRef* parameter is modified.

If the **NWDPASModifyAttrValue** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00100004L NWDP_LE_ATR_VALUE_NOT_FOUND
0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPASAddAttrValue

NWDPASMODObjectListCallback

Lists objects in the Managed Object Database (MODB)
Local Servers: N/A
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_atr.h>

N_TYPEDEF_CALLBACK (nint, NWDPASMODObjectListCallback) (
   NWDPAccessorRef            accessorRef,
   nparam                     callerDefinedParam,
   nint                       totalCallsToBeMade,
   nint                       currentCallCount,
   pNWDPASMODObjectListItem   objectPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

callerDefinedParam

(IN) Specifies a caller-defined contextual variable (optional).

totalCallsToBeMade

(IN) Specifies the number of times the **NWDPPsmListMODObjects** function calls the **NWDPASMODObjectListCallback** function (number of objects plus one).

currentCallCount

(IN) Specifies the current call number.

objectPtr

(IN) Points to the NWDPASMODObjectListItem structure containing the object information.

Return Values

0x00000000	N_SUCCESS

0xFFFFFFFF	N_FAILURE

Remarks

For the *callerDefinedParam* parameter, this function uses the contextual variable to create the following:

a linked list head into which items can be added

a window handle where items can be displayed

any other data or combination of these two steps needed by the callback routine (optional)

For the *totalCallsToBeMade* parameter, the *objectPtr* parameter is NULL for the last callback.

NCP Calls

None

See Also

NWDPASCreateRefBasedOnSet
NWDPAUtilListAttributes
NWDPPrtListMODObjects

NWDPASReleaseRef

Decrements the usage count of an attribute set reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_atr.h>

N_EXTERN_LIBRARY (nint) NWDPASReleaseRef (
   NWDPAccessorRef   accessorRef,
   pNWDPAttrSetRef   attrSetRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

attrSetRefPtr

(IN) Points to the attribute set reference that is released.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

When the usage count reaches zero, the reference is destroyed and its associated memory is freed.

If the **NWDPASReleaseRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER

NCP Calls

None

See Also

NWDPASCreateRef
NWDPASUseRef

NWDPASRemoveAttribute

Removes an attribute from an attribute set
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_atr.h>

N_EXTERN_LIBRARY (nint) NWDPASRemoveAttribute (
   NWDPAccessorRef   accessorRef,
   NWDPASAVPRef      avpRef,
   pNWDPOid          attributeIdPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

avpRef

(IN) Points to the NWDPASAVPStruct structure identifying the attribute set where the attribute is removed.

attributeIdPtr

(IN) Points to the NWDPOid structure containing the OID representing the attribute to be removed (see the nwdp_wko.ogh file for a list of OIDs).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

Remarks

If the attribute pointer of the *avpRef* parameter already points to the correct attribute, pass NULL for the*attributeIdPtr* parameter.

If the **NWDPASRemoveAttribute** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00100001L NWDP_LE_ATR_NOT_IN_SET

NCP Calls

None

See Also

NWDPASAddAttribute

NWDPASRemoveAttrValue

Removes a value from the specified attribute
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_atr.h>

N_EXTERN_LIBRARY (nint) NWDPASRemoveAttrValue (
   NWDPAccessorRef       accessorRef,
   NWDPASAVPRef          avpRef,
   pNWDPOid              attributeIdPtr,
   pNWDPAttributeValue   valuePtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

avpRef

(IN) Points to the NWDPASAVPStruct structure identifying the attribute set from which the value is removed.

attributeIdPtr

(IN) Points to the NWDPOid structure containing the OID representing the attribute from which the value is removed (see the nwdp_wko.ogh file for a list of OIDs).

valuePtr

(IN) Points to the NWDPAttributeValue structure containing the value to be removed.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFF	N_FAILURE

0xFFFFFFFF	N_FAILURE

Remarks

If the attribute pointer of the *avpRef* parameter already points to the correct attribute, pass NULL for the*attributeIdPtr* parameter.

If the value pointer of the *avpRef* parameter already points to the correct value, pass NULL for the *valuePtr* parameter. The value identified by the value pointer of the *avpRef* parameter is removed.

If the **NWDPASRemoveAttrValue** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00100001L NWDP_LE_ATR_NOT_IN_SET
0x00100004L NWDP_LE_ATR_VALUE_NOT_FOUND

NCP Calls

None

See Also

NWDPASAddAttrValue
NWDPASModifyAttrValue

NWDPASSetAVPByAttributeId

Sets the attribute and value pointers of the NWDPASAVPRef to the specified attribute

Local Servers: nonblocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_atr.h>

N_EXTERN_LIBRARY (nint) NWDPASSetAVPByAttributeId (
   NWDPAccessorRef   accessorRef,
   NWDPASAVPRef      avpRef,
   pNWDPOid          attributeIdPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

avpRef

(IN) Points to the NWDPASAVPStruct structure identifying the attribute set.

attributeIdPtr

(IN) Points to the NWDPOid structure containing the OID identifying the attribute to which the pointers of the *avpRef* parameter are set (see the nwdp_wko.ogh file for a list of OIDs).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The attribute and value pointers of the *avpRef* parameter are set to the attribute identified by the *attributeIdPtr* parameter. The value pointer of the *avpRef* parameter is set to the first value of the attribute, or NULL if no values exist.

If the **NWDPASSetAVPByAttributeID** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x0010000L NWDP_LE_ATR_NOT_IN_SET

NCP Calls

None

NWDPASSetModifyOperators

Set the modify flag for attribute(s)
Local Servers: nonblocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_atr.h>

N_EXTERN_LIBRARY (nint) NWDPASSetModifyOperators (
   NWDPAccessorRef          accessorRef,
   NWDPASAVPRef             avpRef,
   NWDPModifyOperatorEnum   operation,
   nbool                    applyToAll);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

avpRef

(IN) Points to the NWDPASAVPStruct structure identifying the attribute set where this operation occurs.

operation

(IN) Specifies the modify type that is set for each attribute.

applyToAll

(IN) Passes N_TRUE or N_FALSE (if true, the modify operator for each attribute in the set is changed; otherwise, only the modify operator of the attribute identified by the attribute pointer of the *avpRef* parameter is changed).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFF	N_FAILURE

0xFFFFFFFF	N_FAILURE

Remarks

All attributes in the set must have an associated modify operator when modifying an object in NDS or the MODB.

The operation values are the following:

NWDP_MODIFY_OP_REPLACE

NWDP_MODIFY_OP_ADD_VALUES

NWDP_MODIFY_OP_REMOVE_VALUES

NWDP_MODIFY_OP_SET_TO_DEFAULT

NWDP_MODIFY_OP_REMOVE_ATTRIBUTE

NCP Calls

None

See Also

NWDPASAddAttribute
NWDPDocModifyAttrs
NWDPJobModifyAttrs
NWDPPrtModifyAttrs
NWDPPrtModifyMODObjectAttrs
NWDPPsmModifyMODObjectAttrs

NWDPASUseRef

Increments the usage count for an attribute set reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_atr.h>

N_EXTERN_LIBRARY (nint) NWDPASUseRef (
   NWDPAccessorRef   accessorRef,
   NWDPAttrSetRef    attrSetRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

attrSetRef

(IN) Specifies the attribute set reference having an incremented usage count.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPASUseRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER

NCP Calls

None

See Also

NWDPASCreateRef
NWDPASReleaseRef

NWDPASValidateRef

Validates an attribute set reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_atr.h>

N_EXTERN_LIBRARY (nint) NWDPASValidateRef (
   NWDPAccessorRef   accessorRef,
   NWDPAttrSetRef    attrSetRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

attrSetRef

(IN) Specifies the attribute set reference to validate.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The **NWDPASValidateRef** function verifies that the attribute set reference is associated with the *accessorRef* parameter when the application has received it from an untrusted third party.

If the **NWDPASValidateRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER

NCP Calls

None

NWDPAUtilAddAttributeWithValues

Adds an attribute containing values to an attribute set
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_aut.h>

N_EXTERN_LIBRARY (nint) NWDPAUtilAddAttributeWithValues (
   NWDPAccessorRef   accessorRef,
   NWDPASAVPRef      destAvpRef,
   pNWDPAttribute    srcAttributePtr,
   pnuint32          qualifierPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

destAvpRef

(IN) Points to the NWDPASAVPStruct structure identifying the attribute set where the attribute is added.

srcAttributePtr

(IN) Points to the NWDPAttribute structure identifying the attribute and values added to the attribute set.

qualifierPtr

(IN) Points to the attribute qualifier (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the attribute is already in the attribute set, its values are removed. The attribute pointer of the *avpRef* parameter is set to the new attribute. The value pointer of the *avpRef* parameter is set to the first value of the new attribute.

If the attribute set is used for MODB or NDS modifications, you can set the modify operator for each attribute now, rather than when you perform the modify call. You can pass a pointer to one of the enumerations from the NWDPModifyOperatorEnum structure for the *qualifierPtr* parameter.

If the **NWDPAUtilAddAttributeWithValues** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPASAddAttribute
NWDPASRemoveAttribute

NWDPAUtilAppendAttrSet

Appends one attribute set to another attribute set
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_aut.h>

N_EXTERN_LIBRARY (nint) NWDPAUtilAppendAttrSet (
   NWDPAccessorRef     accessorRef,
   NWDPASAVPRef        destAvpRef,
   pNWDPAttributeSet   srcAttrSetPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

destAvpRef

(IN) Points to the NWDPASAVPStruct structure identifying the destination attribute set.

srcAttrSetPtr

(IN) Points to the NWDPAttributeSet structure identifying the source attribute set.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The attributes from the source set append to the destination set. Duplicate attributes are not removed.

The attribute and value pointers of the *destAvpRef* parameter are set to the first attribute and value in the destination attribute set.

If the **NWDPAUtilAppendAttrSet** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPAUtilMergeAttrSet

NWDPAUtilAppendAttrValueSet

Appends an attribute value set to another attribute value set
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_aut.h>

N_EXTERN_LIBRARY (nint) NWDPAUtilAppendAttrValueSet (
   NWDPAccessorRef          accessorRef,
   pNWDPAttributeValueSet   destAttrValueSetPtr,
   pNWDPAttributeValueSet   srcAttrValueSetPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

destAttrValueSetPtr

(IN) Points to the NWDPAttributeValueSet structure identifying the destination attribute value set.

srcAttrValueSetPtr

(IN) Points to the NWDPAttributeValueSet structure identifying the source attribute value set.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The values from the source attribute value set are appended to the destination set. Duplicate values are not removed.

If the **NWDPAUtilAppendAttrValueSet** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

NWDPAUtilAsciiToCardinal64

Converts ASCII string to cardinal64 and places it in an attribute value
Local Servers: nonblocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_aut.h>

N_EXTERN_LIBRARY (nint) NWDPAUtilAsciiToCardinal64 (
   NWDPAccessorRef       accessorRef,
   pnstr                 numericDecimalStringPtr,
   pNWDPAttributeValue   valuePtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

numericDecimalStringPtr

(IN) Points to the ASCII string to be converted to cardinal64.

valuePtr

(OUT) Points to the NWDPAttributeValue structure having the value type NWDP_AVT_CARDINAL_64.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

NCP Calls

None

See Also

NWDPAUtilCardinal64ToAscii

NWDPAUtilAttributeListCallback

Lists attributes
Local Servers: N/A
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_aut.h>

N_TYPEDEF_CALLBACK (nint, NWDPAUtilAttributeListCallback) (
   NWDPAccessorRef   accessorRef,
   nparam            callerDefinedParam,
   nint              totalCallsToBeMade,
   nint              currentCallCount,
   pNWDPAttribute    attributePtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

callerDefinedParam

(IN) Specifies a caller-defined contextual variable (optional).

totalCallsToBeMade

(IN) Specifies the number of times the **NWDPAUtilListAttributes** function calls the **NWDPAUtilAttributeListCallback** function (the number of attributes plus one).

currentCallCount

(IN) Specifies the current call number.

attributePtr

(IN) Points to the NWDPAttribute structure.

Return Values

0x00000000	N_SUCCESS
0xFFFFFF	N_FAILURE

FF	

Remarks

For the *callerDefinedParam* parameter, this function uses the contextual variable to create the following:

a linked list head into which items can be added

a window handle where items can be displayed

any other data or combination of these two steps needed by the callback routine (optional)

For the *totalCallsToBeMade* parameter, the *attributePtr* parameter is NULL for the last callback.

NCP Calls

None

See Also

NWDPAUtilListAttributes
NWDPAUtilListAttrValues

NWDPAUtilAttrValueListCallback

Lists attribute values
Local Servers: N/A
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_aut.h>

N_TYPEDEF_CALLBACK (nint, NWDPAUtilAttrValueListCallback) (
   NWDPAccessorRef       accessorRef,
   nparam                callerDefinedParam,
   nint                  totalCallsToBeMade,
   nint                  currentCallCount,
   pNWDPAttributeValue   valuePtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

callerDefinedParam

(IN) Specifies a caller-defined contextual variable (optional).

totalCallsToBeMade

(IN) Specifies the number of times the **NWDPAUtilListAttrValues** function calls the **NWDPAUtilAttrValueListCallback** function (the number of values plus one).

currentCallCount

(IN) Specifies the current call number.

valuePtr

(IN) Points to the NWDPAttributeValue structure.

Return Values

0x00000000	N_SUCCESS
0xFFFFFF	N_FAILURE

FF	

Remarks

For the *callerDefinedParam* parameter, this function uses the contextual variable to create the following:

a linked list head into which items can be added

a window handle where items can be displayed

any other data or combination of these two steps needed by the callback routine (optional)

For the *totalCallsToBeMade* parameter, the *valuePtr* parameter is NULL for the last callback.

NCP Calls

None

See Also

NWDPAUtilListAttributes
NWDPAUtilListAttrValues

NWDPAUtilCardinal64ToAscii

Converts a cardinal64 attribute value to an ASCII string
Local Servers: nonblocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_aut.h>

N_EXTERN_LIBRARY (nint) NWDPAUtilCardinal64ToAscii (
   NWDPAccessorRef       accessorRef,
   pNWDPAttributeValue   valuePtr,
   nbool                 useSeparators,
   nuint                 sizeOfStringBuffer,
   pnstr                 stringBufferPtr,
   pnuint                sizeOfResultPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

valuePtr

(IN) Points to the NWDPAttributeValue structure having the value type NWDP_AVT_CARDINAL_64.

useSeparators

(IN) Specifies the use of separators (if N_TRUE, the string is returned with separators based on the current locality).

sizeOfStringBuffer

(IN) Specifies the size of the buffer containing the resulting ASCII string.

stringBufferPtr

(OUT) Points to the resulting ASCII string.

sizeOfResultPtr

(OUT) Points to the actual size copied into the *stringBufferPtr* parameter (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

To determine the exact amount of memory needed for the *stringBufferPtr* parameter, pass a zero for the *sizeOfStringBuffer* parameter. The NWDP_LE_RESULT_BUFFER_TOO_SMALL error is returned with the *sizeOfResultPtr* parameter identifying the amount of memory needed.

If the **NWDPUtilCardinal64ToAscii** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x000A0006L NWDP_LE_RESULT_BUFFER_TOO_SMALL

NCP Calls

None

See Also

NWDPAUtilAsciiToCardinal64

NWDPAUtilCompareAttrValue

Compares two attribute values
Local Servers: nonblocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_aut.h>

N_EXTERN_LIBRARY (nint) NWDPAUtilCompareAttrValue (
   NWDPAccessorRef       accessorRef,
   pNWDPAttributeValue   attrValue1Ptr,
   pNWDPAttributeValue   attrValue2Ptr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

attrValue1Ptr

(IN) Points to the NWDPAttributeValue structure identifying the first value.

attrValue2Ptr

(IN) Points to the NWDPAttributeValue structure identifying the second value.

Return Values

0x00000000	N_SUCCESS
0x00000001	NWDP_RC_WARNING
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the two attribute values are equal, N_SUCCESS is returned.

If they are not equal, NWDP_RC_WARNING is returned.

If NWDP_RC_WARNING is returned and the *libError* field of the *accessorRef* parameter has the NWDP_EC_CATEGORY_MASK bit set, the *otherError* field of the *accessorRef* parameter reflects the value of the *attrValue1ptr* parameter (relative to the *attrValue2ptr* parameter). This allows sorting to be done.

If the **NWDPAUtilCompareAttrValue** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00100003L NWDP_LE_ATR_NOT_SUPPORTED

NCP Calls

None

See Also

NWDPAUtilCompareDataType

NWDPAUtilCompareDataType

Compares two data types
Local Servers: nonblocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_aut.h>

N_EXTERN_LIBRARY (nint) NWDPAUtilCompareDataType (
   NWDPAccessorRef   accessorRef,
   NWDPAVTEnum       dataType,
   nptr              data1Ptr,
   nptr              data2Ptr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

dataType

(IN) Specifies the structure type of the data being compared (it must be an attribute value type).

data1Ptr

(IN) Points to a structure of the *dataType* parameter.

data2Ptr

(IN) Points to a structure of the *dataType* parameter.

Return Values

0x00000000	N_SUCCESS
0x00000001	NWDP_RC_WARNING
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFF	N_FAILURE

FF	

Remarks

If the compare is equal, N_SUCCESS is returned.

If the compare is not equal, NWDP_RC_WARNING is returned.

If NWDP_RC_WARNING is returned and the *libError* field of the *accessorRef* parameter has the NWDP_EC_CATEGORY_MASK bit set, the *otherError* field of the *accessorRef* parameter reflects the value of the *data1Ptr* parameter (relative to the value of the *data2Ptr* parameter). This allows sorting to be done.

If the **NWDPAUtilCompareDataType** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00100003L NWDP_LE_ATR_NOT_SUPPORTED
0x01000002L NWDP_EC_PARAM_VAL_UNRECOGNIZED

NCP Calls

None

See Also

NWDPAUtilCompareAttrValue

NWDPAUtilDupAttribute

Duplicates an attribute
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_aut.h>

N_EXTERN_LIBRARY (nint) NWDPAUtilDupAttribute (
   NWDPAccessorRef   accessorRef,
   pNWDPAttribute    destAttributePtr,
   pNWDPAttribute    srcAttributePtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

destAttributePtr

(OUT) Points to the NWDPAttribute structure identifying the destination attribute.

srcAttributePtr

(IN) Points to the NWDPAttribute structure identifying the source attribute.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

Call the **NWDPAUtilFreeAttribute** function to free memory allocated by the **NWDPAUtilDupAttribute** function.

If the **NWDPAUtilDupAttribute** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPAUtilDupAttrValue
NWDPAUtilDupDataType
NWDPAUtilFreeAttribute

NWDPAUtilDupAttrValue

Duplicates an attribute value
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_aut.h>

N_EXTERN_LIBRARY (nint) NWDPAUtilDupAttrValue (
   NWDPAccessorRef       accessorRef,
   pNWDPAttributeValue   destAttrValuePtr,
   pNWDPAttributeValue   srcAttrValuePtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

destAttrValuePtr

(OUT) Points to the NWDPAttributeValue structure identifying the destination attribute value.

srcAttrValuePtr

(IN) Points to the NWDPAttributeValue structure identifying the source attribute value.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

Call the **NWDPAUtilFreeAttrValue** function to free all memory allocated by the **NWDPAUtilDupAttrValue** function.

If the **NWDPAUtilDupAttrValue** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPAUtilDupAttribute
NWDPAUtilDupDataType
NWDPAUtilFreeAttrValue

NWDPAUtilDupDataType

Duplicates one of the data types
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_aut.h>

N_EXTERN_LIBRARY (nint) NWDPAUtilDupDataType (
   NWDPAccessorRef   accessorRef,
   NWDPAVTEnum       dataType,
   nptr              destPtr,
   nptr              srcPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

dataType

(IN) Specifies the structure type of the data duplicated (it has to be one of the attribute value types).

destPtr

(IN) Points to the destination structure of a *dataType* parameter.

srcPtr

(IN) Points to the source structure of a *dataType* parameter.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The destination data type structure must already exist. For example, if **NWDP_AVT_OBJECT_IDENTIFICATION** was passed for the *dataType* parameter, pass the address of the NWDPObjectIdentification structure for the *destPtr* parameter. The library allocates memory for all substructures.

Call the **NWDPAUtilFreeDataType** function to free all memory allocated by the **NWDPAUtilDupDataType** function.

If the **NWDPAUtilDupDataType** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00100003L NWDP_LE_ATR_NOT_SUPPORTED
0x01000002L NWDP_EC_PARAM_VAL_UNRECOGNIZED
0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPAUtilDupAttribute
NWDPAUtilDupAttrValue
NWDPAUtilFreeDataType

NWDPAUtilFreeAttribute

Frees all memory associated with an attribute
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_aut.h>

N_EXTERN_LIBRARY (nint) NWDPAUtilFreeAttribute (
   NWDPAccessorRef   accessorRef,
   pNWDPAttribute    attributePtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

attributePtr

(IN) Points to the NWDPAttribute structure identifying the freed attribute.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The memory allocated by the **NWDPAUtilDupAttribute** function is freed. However, the NWDPAttribute structure is caller-owned and is not freed by the library.

NCP Calls

None

See Also

NWDPAUtilDupAttribute

NWDPAUtilFreeAttrValue

Frees all memory associated with an attribute value
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_aut.h>

N_EXTERN_LIBRARY (nint) NWDPAUtilFreeAttrValue (
   NWDPAccessorRef       accessorRef,
   pNWDPAttributeValue   attrValuePtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

attrValuePtr

(IN) Points to the NWDPAttributeValue structure identifying the freed attribute.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

All memory allocated by the **NWDPAUtilDupAttrValue** function is freed. However, the NWDPAttributeValue structure is caller-owned and is not freed by the library.

NCP Calls

None

See Also

NWDPAUtilDupAttrValue

NWDPAUtilFreeAttrValueSet

Frees a list of attribute values
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_aut.h>

N_EXTERN_LIBRARY (nint) NWDPAUtilFreeAttrValueSet (
   NWDPAccessorRef          accessorRef,
   NWDPAttributeValueSet    attrValueSetPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

attrValueSetPtr

(IN) Points to the NWDPAttributeValueSet structure identifying the list of freed attribute values.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *attrValueSetPtr* parameter points to a NWDPAttributeValueSet structure. The *itemPtr* member of this structure is freed. The *itemPtr* member is set to NULL and the *itemCount* member is set to zero.

NCP Calls

None

NWDPAUtilFreeDataType

Frees all memory associated with the specified data type
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_aut.h>

N_EXTERN_LIBRARY (nint) NWDPAUtilFreeDataType (
   NWDPAccessorRef   accessorRef,
   NWDPAVTEnum       dataType,
   nptr              dataPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

dataType

(IN) Specifies the structure type of the freed data.

dataPtr

(IN) Points to the structure of the*dataType* parameter that is freed.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

All memory allocated by the **NWDPAUtilDupDataType** function is freed. However, the data type structure is caller-owned and is not freed

freed. However, the data type structure is caller-owned and is not freed by the library.

NCP Calls

None

See Also

NWDPAUtilDupDataType

NWDPAUtilListAttributes

Lists the attributes in an attribute set
Local Servers: nonblocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_aut.h>

N_EXTERN_LIBRARY (nint) NWDPAUtilListAttributes (
   NWDPAccessorRef                  accessorRef,
   pNWDPAttributeSet                attributeSetPtr,
   NWDPAUtilAttributeListCallback   listCallback,
   nparam                            callerDefinedParam);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

attributeSetPtr

(IN) Points to the NWDPAttributeSet structure containing the attributes to list.

listCallback

(IN) Specifies the application-supplied callback function which is called by the **NWDPAUtilListAttributes** function for each attribute.

callerDefinedParam

(IN) Specifies a caller-defined parameter, containing application-specific data, that is passed to the callback function (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFF	N_FAILURE

0xFFFFFFFF	N_FAILURE

Remarks

The **NWDPAUtilListAttributes** function is generally nonblocking, except when the associated callback function is blocking, then the **NWDPAUtilListAttributes** function is also blocking.

If the **NWDPAUtilListAttributes** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x000A0005L NWDP_LE_UNKNOWN_CALLBACK_ERR

NCP Calls

None

See Also

NWDPAUtilListAttrValues

NWDPAUtilListAttrValues

Lists the values for an attribute
Local Servers: nonblocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_aut.h>

N_EXTERN_LIBRARY (nint) NWDPAUtilListAttrValues (
   NWDPAccessorRef                  accessorRef,
   pNWDPAttribute                   attributePtr,
   NWDPAUtilAttrValueListCallback   listCallback,
   nparam                            callerDefinedParam);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

attributePtr

(IN) Points to the NWDPAttribute structure containing the values to list.

listCallback

(IN) Specifies the application-supplied callback function which is called by the **NWDPAUtilListAttrValues** function for each value.

callerDefinedParam

(IN) Specifies a caller-defined parameter, containing application-specific data, that is passed to the callback function (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFF	N_FAILURE

0xFFFFFFFF	N_FAILURE

Remarks

The **NWDPAUtilListAttrValues** function is generally nonblocking, except when the associated callback function is blocking, then the **NWDPAUtilListAttrValues** function is also blocking.

If the **NWDPAUtilListAttrValues** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x000A0005L NWDP_LE_UNKNOWN_CALLBACK_ERR

NCP Calls

None

See Also

NWDPAUtilListAttributes

NWDPAUtilMergeAttrSet

Merges an attribute set with another attribute set
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_aut.h>

N_EXTERN_LIBRARY (nint) NWDPAUtilMergeAttrSet (
   NWDPAccessorRef     accessorRef,
   NWDPASAVPRef        destAvpRef,
   pNWDPAttributeSet   srcAttrSetPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

destAvpRef

(IN) Points to the NWDPASAVPStruct structure identifying the destination attribute set.

srcAttrSetPtr

(IN) Points to the NWDPAttributeSet structure identifying the source attribute set.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The attributes and values from the source set are placed in the destination set. If an attribute is already in the destination set, the values of the attribute are replaced with those from the source set; however, its qualifier remains the same.

The attribute and value pointers of the *destAvpRef* parameter are set to the first attribute and value in the destination attribute set.

If the **NWDPAUtilListAttrValues** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPAUtilAppendAttrSet

NWDPAUtilRemoveAllAttributes

Removes all attributes from an attribute set
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_aut.h>

N_EXTERN_LIBRARY (nint) NWDPAUtilRemoveAllAttributes (
   NWDPAccessorRef     accessorRef,
   pNWDPAttributeSet   attrSetPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

attrSetPtr

(IN) Points to the NWDPAttributeSet structure identifying the attributes to be removed.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The attribute value set and the attribute ID are freed for each attribute. The *attrSetPtr* parameter points to a NWDPAttributeSet structure. The *itemPtr* member of this structure is not freed. The *itemCount* member is set to zero.

The **NWDPAUtilRemoveAllAttributes** function enables the user to reuse an attribute set without having to call the **NWDPASCreateRef** function.

NCP Calls

None

NWDPBrkCreateRefBasedOnFQN

Creates a broker reference based on the Fully Qualified Name (FQN)
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_brk.h>

N_EXTERN_LIBRARY (nint) NWDPBrkCreateRefBasedOnFQN (
   NWDPAccessorRef   accessorRef,
   pNWDPNSrvFQN      brokerFqnPtr,
   pNWDPBrkRef       brokerRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

brokerFqnPtr

(IN) Points to the NWDPNSrvFQN structure containing the distinguished name of the broker object.

brokerRefPtr

(OUT) Points to the NWDPBrkRef, the resulting broker reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPBrkCreateRefBasedOnFQN** function returns N_FAILURE,

If the **NWDPBrkCreateRefBasedOnFQN** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000005L NWDP_EC_NDS
0x030A0001L NWDP_EC_NO_MEMORY
0x000A000BL NWDP_LE_FQN_NOT_NDPS_BROKER

NCP Calls

None

See Also

NWDPBrkDestroyRef

NWDPBrkDestroyRef

Destroys a broker reference and frees all associated memory
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_brk.h>

N_EXTERN_LIBRARY (nint) NWDPBrkDestroyRef (
   NWDPAccessorRef   accessorRef,
   pNWDPBrkRef       brokerRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

brokerRefPtr

(IN) Points to the destroyed broker reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPBrkDestroyRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER

NCP Calls

None

See Also

NWDPBrkCreateRefBasedOnFQN

NWDPBrkDisableService

Disables a broker service
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_brk.h>

N_EXTERN_LIBRARY (nint) NWDPBrkDisableService (
   NWDPAccessorRef       accessorRef,
   NWDPBrkRef            brokerRef,
   NWDPServiceTypeEnum   serviceType);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

brokerRef

(IN) Specifies the broker reference.

serviceType

(IN) Specifies the disabled service:

NWDP_SERVICE_TYPE_SRS
NWDP_SERVICE_TYPE_ENS
NWDP_SERVICE_TYPE_RMS

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the broker is down, the library sets the necessary NDS attributes so the selected services are disabled when the broker is loaded.

If the service is already disabled, N_SUCCESS is still returned.

If the **NWDPBrkDisableService** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000005L NWDP_EC_NDS
0x03060001L NWDP_EC_BROKER_RESULT

NCP Calls

None

See Also

NWDPBrkEnableService

NWDPBrkEnableService

Enables a broker service
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_brk.h>

N_EXTERN_LIBRARY (nint) NWDPBrkEnableService (
   NWDPAccessorRef       accessorRef,
   NWDPBrkRef            brokerRef,
   NWDPServiceTypeEnum   serviceType);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

brokerRef

(IN) Specifies the broker reference.

serviceType

(IN) Specifies the enabled service:

NWDP_SERVICE_TYPE_SRS
NWDP_SERVICE_TYPE_ENS
NWDP_SERVICE_TYPE_RMS

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the broker is down, the library sets the necessary NDS attributes so the selected services are enabled when the broker is loaded.

If the service is already enabled, N_SUCCESS is still returned.

If the **NWDPBrkEnableService** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000005L NWDP_EC_NDS
0x03060001L NWDP_EC_BROKER_RESULT

NCP Calls

None

See Also

NWDPBrkDisableService

NWDPBrkGetConnectionStatus

Checks the broker connection status, otherwise, it creates a new connection
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_brk.h>

N_EXTERN_LIBRARY (nint) NWDPBrkGetConnectionStatus (
   NWDPAccessorRef   accessorRef,
   pNWDPBrkRef       brokerRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

brokerRef

(IN) Specifies the broker reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the *brokerRef* parameter is not bound to the broker, a bind occurs. If the *brokerRef* parameter is already bound, the connection is checked. If the connection is lost, NWDP_LE_BRK_CONNECTION_LOST is returned.

If the **NWDPBrkGetConnectionStatus** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

NWDPLibErrorMac *(accessorRef)* contains the following:

0x00060001L NWDP_LE_BRK_CONNECTION_LOST
0x03060001L NWDP_EC_BROKER_RESULT
0x05180001L NWDP_EC_BRK_BOUND_TO_WRONG_ONE

NCP Calls

None

NWDPBrkGetFQN

Returns the Fully Qualified Name (FQN) of the broker reference
Local Servers: nonblocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_brk.h>

N_EXTERN_LIBRARY (nint) NWDPBrkGetFQN (
   NWDPAccessorRef      accessorRef,
   NWDPBrkRef           brokerRef,
   nuint                sizeOfBuffer,
   pNWDPNSrvFQNBuffer   fqnBufferPtr,
   pnuint               sizeOfResultPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

brokerRef

(IN) Specifies the broker reference.

sizeOfBuffer

(IN) Specifies the size of the buffer pointed to by the *fqnBufferPtr* parameter.

fqnBufferPtr

(OUT) Points to the resulting FQN.

sizeOfResultPtr

(OUT) Points to the data size copied into the *fqnBufferPtr* parameter (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFF	NWDP_RC_INVALID_ACCESSOR

FE	
0xFFFFFFFF	N_FAILURE

Remarks

An alternative method which saves memory is to pass a zero for the *sizeOfBuffer* parameter. The NWDP_LE_RESULT_BUFFER_TOO_SMALL error is returned with the *sizeOfResultPtr* parameter identifying the needed buffer size. Therefore, you need to allocate the amount of memory identified by the *sizeOfResultPtr* parameter, cast the buffer as an NWDPNSrvFQNBuffer, and pass it as the *fqnBufferPtr* parameter.

If the **NWDPBrkGetFQN** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x000A0006L NWDP_LE_RESULT_BUFFER_TOO_SMALL
0x000A0008L NWDP_LE_REF_NOT_CREATED_WITH_FQN

NCP Calls

None

See Also

NWDPBrkCreateRefBasedOnFQN

NWDPBrkListServices

Lists the services known to the broker
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_brk.h>

N_EXTERN_LIBRARY (nint) NWDPBrkListServices (
   NWDPAccessorRef               accessorRef,
   NWDPBrkRef                    brokerRef,
   NWDPListServicesEnum          supportedVsEnabledFilter,
   NWDPBrkServicesListCallback   listCallback,
   nparam                         callerDefinedParam);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

brokerRef

(IN) Specifies the broker reference.

supportedVsEnabledFilter

(IN) Specifies the type of services to list (pass either NWDP_SERVICES_SUPPORTED or NWDP_SERVICES_ENABLED).

listCallback

(IN) Specifies the application-supplied callback function which is called for each service by the **NWDPBrkListServices** function.

callerDefinedParam

(IN) Specifies a caller-defined parameter, containing application-specific data, that is passed to the callback function (optional).

Return Values

0x00000000	N_SUCCESS

0	
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPBrkListServices** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x03060001L NWDP_EC_BROKER_RESULT
0x000A0005L NWDP_LE_UNKNOWN_CALLBACK_ERR

NCP Calls

None

See Also

NWDPBrkCreateRefBasedOnFQN

NWDPBrkServicesListCallback

Lists the broker services
Local Servers: N/A
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_brk.h>

N_TYPEDEF_CALLBACK (nint, NWDPBrkServicesListCallback) (
   NWDPAccessorRef        accessorRef,
   nparam                 callerDefinedParam,
   nint                   totalCallsToBeMade,
   nint                   currentCallCount,
   pNWDPServiceListItem   itemReceivedPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

callerDefinedParam

(IN) Specifies a caller-defined contextual variable (optional).

totalCallsToBeMade

(IN) Specifies the number of times the **NWDPBrkListServices** function calls the **NWDPBrkServicesListCallback** function (number of services plus one).

currentCallCount

(IN) Specifies the current call number.

itemReceivedPtr

(IN) Points to the NWDPServiceListItem structure containing the service information.

Return Values

0x00000000	N_SUCCESS

0xFFFFFFFF	N_FAILURE

Remarks

For the *callerDefinedParam* parameter, this function uses the contextual variable to create the following:

a linked list head into which items can be added

a window handle where items can be displayed

any other data or combination of these two steps needed by the callback routine (optional)

For the *totalCallsToBeMade* parameter, the *itemReceivedPtr* parameter is NULL for the last callback.

NCP Calls

None

See Also

NWDPBrkDisableService
NWDPBrkEnableService
NWDPBrkListServices

NWDPBrkShutdown

Shuts down the broker and unloads the Netware Loadable Module (NLM)
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_brk.h>

N_EXTERN_LIBRARY (nint) NWDPBrkShutdown (
   NWDPAccessorRef   accessorRef,
   pNWDPBrkRef       brokerRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

brokerRef

(IN) Points to the reference of the shutdown broker.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPBrkShutdown** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x03060001L NWDP_EC_BROKER_RESULT

NCP Calls

None

See Also

NWDPBrkCreateRefBasedOnFQN

NWDPBrkValidateRef

Validates a broker reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_brk.h>

N_EXTERN_LIBRARY (nint) NWDPBrkValidateRef (
   NWDPAccessorRef   accessorRef,
   NWDPBrkRef        brokerRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

brokerRef

(IN) Specifies the broker reference to validate.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The **NWDPBrkValidateRef** function verifies that the broker reference is associated with the *accessorRef* parameter when the application has received it from an untrusted party.

If the **NWDPBrkValidateRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER

NCP Calls

None

NWDPDocCancel

Cancels a document within a submitted or unsubmitted job object in the MODB (Managed Object Database)

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_doc.h>

N_EXTERN_LIBRARY (nint) NWDPDocCancel (
   NWDPAccessorRef   accessorRef,
   NWDPDocRef        docRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

docRef

(IN) Specifies the reference to the canceled document object in the MODB.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The state of the job is considered unsubmitted (document data file is open on the server) until the **NWDPJobSubmit** function has been called to submit and close the data file.

The document object in the MODB is removed. If this is the only document, the job object is also removed.

If the **NWDPDocCancel** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00080001L NWDP_LE_DOC_INVALID_OPERATION
0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN

NCP Calls

None

See Also

NWDPJobCancel

NWDPDocCreateRef

Creates a new document reference inside the specified job reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_doc.h>

N_EXTERN_LIBRARY (nint) NWDPDocCreateRef (
   NWDPAccessorRef   accessorRef,
   NWDPJobRef        jobRef,
   nuint             position,
   NWDPAttrSetRef    attrSetRef,
   pNWDPDocRef       docRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

jobRef

(IN) Specifies the job reference to which the new document reference is assigned.

position

(IN) Specifies the position of the document relative to the other documents in the job.

attrSetRef

(IN) Specifies the reference to the attribute set used to create the document object in the MODB (optional).

docRefPtr

(OUT) Points to the NWDPDocRef, the resulting document reference.

Return Values

0x00000000	N_SUCCESS

0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the *jobRef* parameter is already associated with a job object in the MODB, that object must be in an unsubmitted state. The **NWDPJobSubmit** function should not have been previously called with this job reference.

For the *position* parameter, a value of zero represents the first position, one the second, and so forth. You can pass NWDP_DOC_POSITION_LAST for the last position.

The *attrSetRef* parameter can be NULL if the default values from the printer agent are preferred.

If the *attrSetRef* parameter is non-NULL, it must not be released until the job reference is submitted.

If the **NWDPDocCreateRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00090001L NWDP_LE_JOB_INVALID_OPERATION
0x00090002L NWDP_LE_JOB_BAD_DOC_POSITION
0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPDocDestroyRef
NWDPDocTransferFile

NWDPDocWriteBuf

NWDPJobSubmit

NWDPDocCreateRefBasedOnDocId

Creates a document reference for a specific document ID
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_doc.h>

N_EXTERN_LIBRARY (nint) NWDPDocCreateRefBasedOnDocId (
   NWDPAccessorRef   accessorRef,
   NWDPJobRef        jobRef,
   nuint32           docId,
   pNWDPDocRef       docRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

jobRef

(IN) Specifies the job reference to which the new document reference is assigned.

docId

(IN) Specifies the document identifier of the document object in the MODB.

docRefPtr

(OUT) Points to the NWDPDocRef, the resulting document reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The job reference should be associated with a submitted job object.

If the **NWDPDocCreateRefBasedOnDocId** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPDocDestroyRef
NWDPJobCreateRefBasedOnJobId

NWDPDocDestroyRef

Destroys a document reference and frees all associated memory
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_doc.h>

N_EXTERN_LIBRARY (nint) NWDPDocDestroyRef (
   NWDPAccessorRef   accessorRef,
   pNWDPDocRef       docRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

docRefPtr

(IN) Points to the document reference that is destroyed.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the job object of the document is submitted, the job and document objects are not affected. If the job object of the document is unsubmitted (the **NWDPJobSubmit** function has not been called), the document object is removed from the MODB. If the document object is the only document, the job object is also removed.

If the **NWDPDocDestroyRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER

NCP Calls

None

See Also

NWDPDocCreateRef
NWDPDocCreateRefBasedOnDocId

NWDPDocGetAttributeSet

Returns the attribute set of the referenced document
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_doc.h>

N_EXTERN_LIBRARY (nint) NWDPDocGetAttributeSet (
   NWDPAccessorRef   accessorRef,
   NWDPDocRef        docRef,
   NWDPOidSetRef     requestedOidSetRef,
   pNWDPAttrSetRef   resultAttrSetRefPtr,
   pNWDPASAVPRef     avpRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

docRef

(IN) Specifies the reference to the MODB document object for which the attribute set is returned.

requestedOidSetRef

(IN) Specifies the set of attributes that are returned for the document object (pass NULL to return all of the attributes for this object).

resultAttrSetRefPtr

(OUT) Points to the NWDPAttrSetRef, the resulting attribute set reference of the document object.

avpRefPtr

(OUT) Points to the NWDPASAVPRef, the resulting attribute and value pointer reference (optional).

Return Values

0x00000000	N_SUCCESS

0	
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *requestedOidSetRef* parameter should contain the attribute identifier for each attribute preferred. See nwdp_wko.ogh file for a list of these OIDs.

Call the **NWDPASReleaseRef** function to free allocated memory.

The attribute and value pointers of the *avpRefPtr* parameter are set to the first attribute and value in the attribute set of the document.

If the **NWDPDocGetAttributeSet** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00080004L NWDP_LE_DOC_NO_SUCH_DOC
0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPASListAttributes
NWDPASReleaseRef
NWDPDocCreateRef
NWDPOSCreateRef

NWDPDocGetId

Retrieves the document identifier
Local Servers: nonblocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_doc.h>

N_EXTERN_LIBRARY (nint) NWDPDocGetId (
   NWDPAccessorRef   accessorRef,
   NWDPDocRef        docRef,
   pnuint32          docIdPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

docRef

(IN) Specifies the reference to the MODB document object for which the identifier is returned.

docIdPtr

(OUT) Points to the resulting document identifier.

Return Values

0x00000000	N_SUCCESS
0x00000001	NWDP_RC_WARNING
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

A document identifier cannot be returned until the job object has been submitted by calling the **NWDPJobSubmit** function.

If the **NWDPDocGetId** function returns NWDP_RC_WARNING, the NWDPLibErrorMac *(accessorRef)* contains the following;

0x00080001L NWDP_LE_DOC_UNVALIDATED_DOC_ID

If the **NWDPDocGetId** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00080001L NWDP_LE_DOC_INVALID_OPERATION

NCP Calls

None

See Also

NWDPJobSubmit

NWDPDocModifyAttrs

Modifies attributes of a document object in the MODB
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_doc.h>

N_EXTERN_LIBRARY (nint) NWDPDocModifyAttrs (
   NWDPAccessorRef          accessorRef,
   NWDPDocRef               docRef,
   NWDPAttrSetRef           modifyRef,
   NWDPModifyOperatorEnum   docOperator);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

docRef

(IN) Specifies the reference to the MODB document object whose attributes are modified.

modifyRef

(IN) Specifies the reference to the attribute set used to modify the document object.

docOperator

(IN) Specifies the modify operator applied to all attributes identified by the *modifyRef* parameter.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

FF	

Remarks

Each attribute in the attribute set specified by the *modifyRef* parameter has a modify operator assigned to it. This modify operator defines the modification to occur for that attribute. This value can be set for each attribute using the *qualifierPtr* parameter of the **NWDPASAddAttribute** function. Or, you can overwrite this value for the entire set by passing a non-NULL (other than NWDP_MODIFY_OP_NULL) value for the *docOperator* parameter.

If the **NWDPDocModifyAttrs** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00080004L NWDP_LE_DOC_NO_SUCH_DOC
0x00090005L NWDP_LE_JOB_NO_SUCH_JOB
0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN

NCP Calls

None

See Also

NWDPASAddAttribute
NWDPASCreateRef
NWDPDocCreateRefBasedOnDocId
NWDPJobModifyAttrs

NWDPDocTransferFile

Creates a document object and copies data to that object
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_doc.h>

N_EXTERN_LIBRARY (nint) NWDPDocTransferFile (
   NWDPAccessorRef               accessorRef,
   NWDPDocRef                    docRef,
   pnstr16                       sourceFilePath16Ptr,
   NWDPDocTransferFileCallback   progressFunc,
   nparam                         callerDefinedParam);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

docRef

(IN) Specifies the document reference associated with the new document object.

sourceFilePath16Ptr

(IN) Points to the Unicode string that describes the location of the source file that is transferred to the printer.

progressFunc

(IN) Specifies the application-supplied callback function called by the **NWDPDocTransferFile** function to report the progress of the transfer of the data (optional).

callerDefinedParam

(IN) Specifies a caller-defined parameter, containing application-specific data, that is passed to the callback function (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *docRef* parameter identifies the job and the printer agent of the document object. The job and the document objects are created by the printer agent during this function. The optional attribute sets associated with the job and document references are added to these objects. The document object created in the MDOB is in the unsubmitted state (document data file is open). Call the **NWDPJobSubmit** function to submit the job and to close the data file.

If the **NWDPDocTransferFile** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00080002L NWDP_LE_DOC_ZERO_LENGTH
0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x030A0001L NWDP_EC_NO_MEMORY
0x000A0005L NWDP_LE_UNKNOWN_CALLBACK_ERR

NCP Calls

None

See Also

NWDPDocCreateRef
NWDPDocWriteBuf
NWDPJobCreateRef
NWDPJobSubmit

NWDPDocTransferFileCallback

Reports the progress of data transfer
Local Servers: N/A
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_doc.h>

N_TYPEDEF_CALLBACK (nint, NWDPDocTransferFileCallback) (
   NWDPAccessorRef   accessorRef,
   nparam            callerDefinedParam,
   nuint32           totalBytesToBeTransferred,
   nuint32           currentNumberTransferred);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

callerDefinedParam

(IN) Specifies a caller-defined contextual variable used by this function (optional).

totalBytesToBeTransferred

(IN) Specifies the number of bytes in the source file.

currentNumberTransferred

(IN) Specifies the current amount of data that has already been transferred.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFF	N_FAILURE

Remarks

For the *callerDefinedParam* parameter, this function uses the contextual variable to create the following:

a linked list head into which items can be added

a window handle where items can be displayed

any other data or combination of these two steps needed by the callback routine (optional)

The *currentNumberTransferred* parameter approaches the byte size of the *totalBytesToBeTransferred* parameter but does not equal it until after all the data has been copied. Screen cleanup and other items occur at this point.

NCP Calls

None

See Also

NWDPDocTransferFile

NWDPDocValidateRef

Validates a document reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPDocValidateRef (
   NWDPAccessorRef   accessorRef,
   NWDPDocRef        docRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

docRef

(IN) Specifies the document reference to validate.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The **NWDPDOCValidateRef** function verifies that the document reference is associated with the *accessorRef* parameter when the application has received it from an untrusted party.

If the **NWDPDocValidateRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER

NCP Calls

None

NWDPDocWriteBuf

Writes data to the document data file
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_doc.h>

N_EXTERN_LIBRARY (nint) NWDPDocWriteBuf (
   NWDPAccessorRef   accessorRef,
   NWDPDocRef        docRef,
   nptr              bufferPtr,
   nuint             sizeOfBuffer);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

docRef

(IN) Specifies the reference to the document object.

bufferPtr

(IN) Points to the buffer containing the data.

sizeOfBuffer

(IN) Specifies the size of the data contained in the *bufferPtr* parameter.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *docRef* parameter identifies the job object and the printer agent of the document object. If the document or job objects do not exist, they are created during this function. The document object in the MODB is in the unsubmitted state (document data file is open). Call the **NWDPJobSubmit** function to submit the job and to close the data file.

If the **NWDPDocWriteBuf** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPDocCreateRef
NWDPDocWriteBuf
NWDPJobCreateRef
NWDPJobSubmit

NWDPFltAppendAttrSetMatch

Appends the filter expression which the *matchFlag* and *avpRef* parameters describe to the existing *filterRef* parameter

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_flt.h>

N_EXTERN_LIBRARY (nint) NWDPFltAppendAttrSetMatch (
   NWDPAccessorRef          accessorRef,
   NWDPFilterRef            filterRef,
   NWDPASAVPRef             avpRef,
   NWDPFilterAttrSetMatch   matchFlag);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

filterRef

(IN) Specifies the NWDPFilterRef containing the filter reference.

avpRef

(IN) Points to the attribute and optional printer values to compare in the NWDPASAVPStruct structure.

matchFlag

(IN) Specifies the expected relationship between the attribute in the object and the accompanying values in the attribute's value set structure.

Return Values

0x00000000	N_SUCCESS

Remarks

Remarks

The *matchFlag* parameter contains the following values:

0	NWDP_FLT_MATCH_EQUAL	Match if the requested attribute value is equal to the one provided.
2	NWDP_FLT_MATCH_GE	Match if the requested attribute value is GREATER_THAN_OR_EQUAL to the one provided.
3	NWDP_FLT_MATCH_LE	Match if the requested attribute value is LESS_THAN_OR_EQUAL to the one provided.
4	NWDP_FLT_MATCH_PRESENT	Match if the requested attribute has any value (its value exists).
5	NWDP_FLT_MATCH_SUBSET	Match if the requested value set is completely contained in the list of values belonging to an attribute.
6	NWDP_FLT_MATCH_SUPERSET	Match if all values are within the requested set.
7	NWDP_FLT_MATCH_NON_DISJOINT	Match if any value is within the requested set.

NCP Calls

None

See Also

NWDPFltAppendDelimiter
NWDPFltAppendSubstringMatch
NWDPFltCreateRef
NWDPFltDestroyRef
NWDPFltValidateRef

NWDPFltAppendDelimiter

Appends a delimiter to an existing filter
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_flt.h>

N_EXTERN_LIBRARY (nint) NWDPFltAppendDelimiter (
   NWDPAccessorRef       accessorRef,
   NWDPFilterRef         filterRef,
   NWDPFilterDelimiter   delimiter);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

filterRef

(IN) Specifies the NWDPFilterRef containing the filter reference.

delimiter

(IN) Specifies the appended delimiter.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *delimiter* parameter can have the following values:

1 NWDP_FLT_DELIM_AND
2 NWDP_FLT_DELIM_OR
3 NWDP_FLT_DELIM_NOT
9 NWDP_FLT_DELIM_LPAREN
10 NWDP_FLT_DELIM_RPAREN
11 NWDP_FLT_DELIM_END

If the **NWDPFltAppendDelimiter** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPFltAppendAttrSetMatch
NWDPFltAppendSubstringMatch
NWDPFltCreateRef
NWDPFltDestroyRef
NWDPFltValidateRef

NWDPFltAppendSubstringMatch

Appends the filter expression to match substrings in the existing filter reference

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_flt.h>

N_EXTERN_LIBRARY (nint) NWDPFltAppendSubstringMatch (
   NWDPAccessorRef   accessorRef,
   NWDPFilterRef     filterRef,
   pNWDPSubStrings   attrSubstringMatchPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

filterRef

(IN) Specifies the NWDPFilterRef containing the filter reference.

attrSubstringMatchPtr

(IN) Points to a substring matching standard.

Return Values

0x00000000	N_SUCCESS

Remarks

Call the **NWDPFltAppendSubstringMatch** function to create partial substring matches for a single attribute with the following syntax:

NWDP_AVT_TEXT
NWDP_AVT_DESCRIPTIVE_NAME
NWDP_AVT_DESCRIPTOR

NWDP_AVT_DESCRIPTOR
NWDP_AVT_MESSAGE
NWDP_AVT_ERROR_MESSAGE
NWDP_AVT_SIMPLE_NAME
NWDP_AVT_DIST_NAME_STR
NWDP_AVT_DISTINGUISHED_NAME
NWDP_AVT_OCTET_STRING
NWDP_AVT_LOCALE
NWDP_AVT_FILE_REFERENCE
NWDP_AVT_PRT_CONTAINED_OBJ_ID

In the NWDPSubStrings structure, a specified *initialValueOption* field can exist, which must be found at the beginning of the string, whereas, the *anyValueSetOption* field matches only if at the end. The *anyValueSetOption* field allows an ordered set of substrings to appear between the optional *initialValueOption* and the *finalValueOption* fields.

Also in the NWDPSubStrings structure, the *matchCriteria* field specifies how to compare individual characters within the following substrings (in progressively greater degrees of precision):

NWDP_MATCH_EXACT
NWDP_MATCH_CASE_INSENSITIVE
NWDP_MATCH_SAME_LETTER
NWDP_MATCH_APPROXIMATE

NCP Calls

None

See Also

NWDPFltAppendAttrSetMatch
NWDPFltAppendDelimiter
NWDPFltCreateRef
NWDPFltDestroyRef
NWDPFltValidateRef

NWDPFltCreateRef

Creates a filter reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_flt.h>

N_EXTERN_LIBRARY (nint) NWDPFltCreateRef (
   NWDPAccessorRef   accessorRef,
   pNWDPFilterRef    filterRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

filterRefPtr

(OUT) Points to the NWDPFilterRef containing the filter reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

Filters limit the number of list objects by providing an algebraic expression. The server utilizes this expression to reduce the selected objects for which to return information. For example, in listing the name service (NDS) printer objects which have Description attribute string "On Bob's Desk" and a state of "ready" or "busy," the pseudo-code for the filter

is the following:

```
(Description .EQ. "On Bob's Desk").AND.(STATE .EQ. "Ready". OR. State .
```

Create a filter listing printers contained in such attributes by making the following calls:

```
nint MakeFilterForBobsActivePrinters (     NWDPAccessorRef      accesso

{

    NWDPAVPRef           avpRef;    NWDPAttributeValue   value;

    /*    // Create an Attribute Set with two attributes that correspon

    if (NWDPASCreateRef(accessorRef, 2, &attrSetRef, &avpRef))       go

    if (NWDPASAddAttribute(accessorRef, avpRef,         pNDPSATT_NDS_DE

    value.designator = NWDP_AVT_TEXT;

    NWDPTextInitMac(value.u.text,L"On Bob's Desk");

    if (NWDPASAddAttrValue(accessorRef, avpRef,         NULL, /* avpRef
 ErrorExit;

    if (NWDPASAddAttribute(accessorRef, avpRef,         pNDPSATT_PRINTE

    /*    // Both possible values are placed on the same attribute

    value.designator = NWDP_AVT_OBJECT_IDENTIFIER;    value.u.identifie
 (pNWDPOid)ID_VAL_STATE_READY;

    if (NWDPASAddAttrValue(accessorRef, avpRef,         NULL, /* avpRef
 ErrorExit;

    /* value.designator = NWDP_AVT_OBJECT_IDENTIFIER; redundant */    v
    value.u.identifier.oidStructPtr  = (pNWDPOid)ID_VAL_STATE_BUSY;

    if (NWDPASAddAttrValue(accessorRef, avpRef,         NULL, /* avpRef
 ErrorExit;

    /*    // Create a filter from the contents of the Attribute Set and

    */    if (NWDPFltCreateRef(accessorRef, &filterRef))       goto Err

    if (NWDPFltAppendDelimiter(accessorRef, filterRef,         NWDP_FLI

    /*    // Position the AVP Ref to the Description for    // ========
 ====================================    */

    if (NWDPASSetAVPByAttributeId(accessorRef, avpRef,         pNDPSATT

       if (NWDPFltAppendAttrSetMatch (accessorRef, filterRef,

    if (NWDPFltAppendDelimiter (accessorRef, filterRef,         NWDP_FL
```

```
    /*    // =========    // = .AND. =    // =========    */

     if (NWDPFltAppendDelimiter( accessorRef, filterRef,         NWDP_F
 ==============================    // = (State .EQ. "Ready","Busy") =

    if (NWDPFltAppendDelimiter (accessorRef, filterRef,         NWDP_FL

    if (NWDPASSetAVPByAttributeId(accessorRef, avpRef,         pNDPSATT

    if (NWDPFltAppendAttrSetMatch (accessorRef, filterRef,          avp

    if (NWDPFltAppendDelimiter (accessorRef, filterRef,         NWDP_FL

    /*    // Wrap up by adding an END    */

    if (NWDPFltAppendDelimiter( accessorRef, filterRef,         NWDP_FL

    /*    // The Attribute Set is no longer needed since the filter hol

    if (NWDPASReleaseRef(accessorRef, &attrSetRef))    {       attrSetR

    // Now return the Filter ref we just made and let the caller destro
    return(N_SUCCESS);ErrorExit:

    /*    // Clean up refs with consumed memory resources freed.    */

    if (attrSetRef != NULL)        NWDPASReleaseRef(accessorRef, &attrS

        NWDPFltDestroyRef(accessorRef, &filterRef);     return(N_FAILUR
```

If the **NWDPFltCreateRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPFltAppendAttrSetMatch
NWDPFltAppendDelimiter
NWDPFltAppendSubstringMatch
NWDPFltDestroyRef
NWDPFltValidateRef

NWDPFltDestroyRef

Destroys a filter reference and frees all associated memory
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_flt.h>

N_EXTERN_LIBRARY (nint) NWDPFltDestroyRef (
   NWDPAccessorRef   accessorRef,
   pNWDPFilterRef    filterRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

filterRefPtr

(OUT) Points to the NWDPFilterRef containing the filter reference to be destroyed.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *filterRefPtr* parameter is not valid after the **NWDPFltDestroyRef** function completes.

If the **NWDPFltDestroyRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

NWDPLibErrorMac *(accessorRef)* contains the following:

0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPFltAppendAttrSetMatch
NWDPFltAppendDelimiter
NWDPFltAppendSubstringMatch
NWDPFltCreateRef
NWDPFltValidateRef

NWDPFltValidateRef

Validates a filter reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_flt.h>

N_EXTERN_LIBRARY (nint) NWDPFltValidateRef (
   NWDPAccessorRef   accessorRef,
   pNWDPFilterRef    filterRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

filterRefPtr

(OUT) Points to the NWDPFilterRef containing the filter reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The **NWDPFltValidateRef** function verifies that the filter reference is associated with the *accessorRef* parameter when the application has received it from an untrusted third party.

The **NWDPFltValidateRef** function does not validate the structures contained in the NWDPFilterRef, instead it only verifies that it was

contained in the NWDPFilterRef, instead it only verifies that it was created using the **NWDPFltCreateRef** function.

If the **NWDPFltValidateRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPFltCreateRef

NWDPJobCancel

Cancels a job object in the Managed Object Database (MODB)
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_job.h>

N_EXTERN_LIBRARY (nint) NWDPJobCancel (
   NWDPAccessorRef   accessorRef,
   pNWDPJobRef       jobRefPtr,
   nuint32           flags,
   nuint32           retentionPeriod);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

jobRefPtr

(IN) Points to the reference of the canceled job object in the MODB.

flags

(IN) Specify whether to change the retention attribute and/or destroy the job reference.

retentionPeriod

(IN) Specifies the time period in seconds the job is retained before it is destroyed.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *flags* parameter can have any one or combination of the following values:

NWDP_JOB_FLG_SET_RETENTION
NWDP_JOB_FLG_DESTORY_REF
NULL

If the retention period attribute of the job is zero, the job object is deleted from the MODB.

NWDP_JOB_FLG_SET_RETENTION must be specified in the *flags* parameter or the *retentionPeriod* parameter has no effect.

The job object can be in the submitted or unsubmitted state. The state of the job is considered unsubmitted (document data file is open) until the **NWDPJobSubmit** function is called to submit and to close the data file.

If the **NWDPJobCancel** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00090001L NWDP_LE_JOB_INVALID_OPERATION
0x00090005L NWDP_LE_JOB_NO_SUCH_JOB
0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN

NCP Calls

None

See Also

NWDPDocCancel

NWDPJobCopy

Copies a job within the same printer agent or between two printer agents
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_job.h>

N_EXTERN_LIBRARY (nint) NWDPJobCopy (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        destPrinterRef,
   NWDPJobRef        srcJobRef,
   pNWDPJobRef       destJobRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

destPrinterRef

(IN) Specifies the reference to the printer that receives the job (pass NULL if the copy is within the same device).

srcJobRef

(IN) Specifies the reference to the copied job object in the MODB.

destJobRefPtr

(OUT) Points to the NWDPJobRef, the resulting job reference (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *srcJobRef* parameter identifies the source printer device. If the source and destination printers are controlled-access printers (represented in NDS), they must reside within the same tree.

If the **NWDPJobCopy** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00090005L NWDP_LE_JOB_NO_SUCH_JOB
0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x030A0001L NWDP_EC_NO_MEMORY
0x000C000DL NWDP_LE_PRT_DEST_TREE_DIFFERS

NCP Calls

None

See Also

NWDPJobMove
NWDPPrtJobCopy
NWDPPrtJobMove

NWDPJobCreateRef

Creates a new job reference contained by the specified printer reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_job.h>

N_EXTERN_LIBRARY (nint) NWDPJobCreateRef (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        printerRef,
   NWDPAttrSetRef    attrSetRef,
   pNWDPJobRef       jobRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the printer reference to which the new job reference is assigned.

attrSetRef

(IN) Specifies the reference to the attribute set used to create the job object in the MODB (optional).

jobRefPtr

(OUT) Points to the NWDPJobRef, the resulting job reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

For the *attrSetRef* parameter, pass NULL if the default values from the printer agent are preferred.

If the *attrSetRef* parameter is non-NULL, it must not be released until the job reference is submitted.

If the **NWDPJobCreateRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPDocCreateRef
NWDPJobDestroyRef

NWDPJobCreateRefBasedOnJobId

Creates a job reference for a specified job identifier
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_job.h>

N_EXTERN_LIBRARY (nint) NWDPJobCreateRefBasedOnJobID (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        printerRef,
   nuint32           jobId,
   pNWDPJobRef       jobRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the printer reference containing the new job reference.

jobId

(IN) Specifies the job identifier of the job object in the MODB.

jobRefPtr

(OUT) Points to the NWDPJobRef, the resulting job reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The job reference should be associated with a submitted job object.

If the **NWDPJobCreateRefBasedOnJobID** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPDocCreateRefBasedOnDocId
NWDPJobDestroyRef

NWDPJobDestroyRef

Destroys a job reference and its associated document references and frees all associated memory

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_job.h>

N_EXTERN_LIBRARY (nint) NWDPJobDestroyRef (
   NWDPAccessorRef   accessorRef,
   pNWDPJobRef       jobRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

jobRefPtr

(IN) Points to the job reference that is destroyed.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the job reference identifies a submitted job object in the MODB, the job object is not affected. If the job object is unsubmitted (the **NWDPJobSubmit** function is not called), the job object is removed from the MODB.

If the **NWDPJobDestroyRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER

NCP Calls

None

See Also

NWDPJobCreateRef
NWDPJobCreateRefBasedOnJobId

NWDPJobGetAttributeSet

Returns the attribute set of the referenced job
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_job.h>

N_EXTERN_LIBRARY (nint) NWDPJobGetAttributeSet (
   NWDPAccessorRef   accessorRef,
   NWDPJobRef        jobRef,
   NWDPOidSetRef     requestedOidSetRef,
   pNWDPAttrSetRef   resultAttrSetRefPtr,
   pNWDPASAVPRef     avpRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

jobRef

(IN) Specifies the reference to the MODB job object for which the attribute set is returned.

requestedOidSetRef

(IN) Specifies the set of attributes returned for the job object (pass NULL to return all of the attributes for the object).

resultAttrSetRefPtr

(OUT) Points to the NWDPAttrSetRef, the resulting attribute set reference of the job object.

avpRefPtr

(OUT) Points to the NWDPASAVPRef, the resulting attribute and value pointer reference (optional).

Return Values

0x00000000	N_SUCCESS

0	
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *requestedOidSetRef* parameter should contain the attribute identifier for each desired attribute. The nwdp_wko.ogh file contains a list of the OIDs.

Call the **NWDPASReleaseRef** function to free allocated memory.

The attribute and value pointers of the *avpRefPtr* parameter are set to the first attribute and value listed in the attribute set of the job.

If the **NWDPJobGetAttributeSet** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00080004L NWDP_LE_JOB_NO_SUCH_JOB
0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPASListAttributes
NWDPASReleaseRef
NWDPJobCreateRef
NWDPOSCreateRef

NWDPJobGetId

Retrieves the job identifier of the referenced job
Local Servers: nonblocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_job.h>

N_EXTERN_LIBRARY (nint) NWDPJobGetId (
   NWDPAccessorRef   accessorRef,
   NWDPJobRef        jobRef,
   pnuint32          jobIdPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

jobRef

(IN) Specifies the reference to the MODB job object for which the identifier is returned.

jobIdPtr

(OUT) Points to the resulting job identifier.

Return Values

0x00000000	N_SUCCESS
0x00000001	NWDP_RC_WARNING
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

A job identifier cannot be returned until the job object has been created (the **NWDPJobSubmit**, **NWDPDocTransferFile** or **NWDPDocWriteBuf** function must have been previously called).

If the **NWDPJobGetId** function returns NWDP_RC_WARNING, the NWDPLibErrorMac *(accessorRef)*contains the following;

0x00090004L NWDP_LE_JOB_UNVALIDATED_JOB_ID

If the **NWDPJobGetId** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00090001L NWDP_LE_JOB_INVALID_OPERATION

NCP Calls

None

See Also

NWDPDocTransferFile
NWDPDocWriteBuf
NWDPJobSubmit

NWDPJobGetStatus

Retrieves the status of the referenced job
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_job.h>

N_EXTERN_LIBRARY (nint) NWDPJobGetStatus (
   NWDPAccessorRef   accessorRef,
   NWDPJobRef        jobRef,
   pNWDPAttrSetRef   resultAttrSetRefPtr,
   pNWDPASAVPRef     avpRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

jobRef

(IN) Specifies the reference to the MODB job object for which status is returned.

resultAttrSetRefPtr

(OUT) Points to the NWDPAttrSetRef, the resulting attribute set reference of the job status attributes.

avpRefPtr

(OUT) Points to the NWDPASAVPRef, the resulting attribute and value pointer reference (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

FF	

Remarks

The attributes returned in the *resultAttrSetRefPtr* parameter are current job state and job state reasons. You can find their attribute identifiers in the nwdp_wko.ogh file as pNDPSATT_CURRENT_JOB_STATE (*accessorRef*) and pNDPSATT_JOB_STATE_REAONS (*accessorRef*). These attributes are queried from the MODB. The List Object Attributes (LOA) does not fail if a particular attribute is not found. Call the **NWDPASSetAVPByAttributeId** function to determine which attributes, if any, are returned.

Call the **NWDPASReleaseRef** function to free allocated memory.

If the **NWDPJobGetStatus** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00080004L NWDP_LE_JOB_NO_SUCH_JOB
0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPASReleaseRef
NWDPASSetAVPByAttributeId

NWDPJobInterrupt

Interrupts the job currently printing to process the referenced job
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_job.h>

N_EXTERN_LIBRARY (nint) NWDPJobInterrupt (
   NWDPAccessorRef   accessorRef,
   NWDPJobRef        jobRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

jobRef

(IN) Specifies the reference to the job object that is being processed.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The interrupting job must be a submitted job. The printer resumes processing the uncompleted job upon completion of the interrupting job.

If the **NWDPJobInterrupt** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00090001L NWDP_LE_JOB_INVALID_OPERATION
0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN

NCP Calls

None

NWDPJobModifyAttrs

Modifies the attributes of a job object or the document object in the Managed Object Database (MODB)

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_job.h>

N_EXTERN_LIBRARY (nint) NWDPJobModifyAttrs (
   NWDPAccessorRef            accessorRef,
   NWDPJobRef                 jobRef,
   NWDPAttrSetRef             jobModifyRef,
   NWDPASModifyOperatorEnum   jobOperator,
   NWDPDocRef                 docRef,
   NWDPAttrSetRef             docModifyRef,
   NWDPASModifyOperatorEnum   docOperator);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

jobRef

(IN) Specifies the reference to the modified job object in the MODB.

jobModifyRef

(IN) Specifies the reference to the attribute set used to modify the job object.

jobOperator

(IN) Specifies the modify operator applied to each attribute in the set specified by the *jobModifyRef* parameter.

docRef

(IN) Specifies the reference to the modified MODB document object: this document reference must be assigned to the *jobRef* parameter (optional).

docModifyRef

(IN) Specifies the reference to the attribute set used to modify the document object (optional).

docOperator

(IN) Specifies the modify operator applied to each attribute in the set as indicated by the *docModifyRef* parameter (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *docModifyRef* parameter must be supplied if the *docRef* parameter is non-NULL.

Each attribute in the attribute sets specified by the *jobModifyRef* and *docModifyRef* parameters has a modify operator assigned to it. This modify operator defines the modification to occur for that attribute. You can set the value of each attribute by using the *qualifierPtr* parameter of the **NWDPASAddAttribute** function. Or, you can overwrite this value for the entire set by passing in a non-NULL (other than NWDP_MODIFY_OP_NULL) value for the *jobOperator* and *docOperator* parameters.

If the **NWDPJobModifyAttrs** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00080004L NWDP_LE_DOC_NO_SUCH_DOC
0x00090005L NWDP_LE_JOB_NO_SUCH_JOB
0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN

NCP Calls

None

See Also

NWDPASAddAttribute
NWDPASCreateRef
NWDPDocModifyAttrs
NWDPJobCreateRefBasedOnJobId

NWDPJobMove

Moves the referenced job to another printer
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_job.h>

N_EXTERN_LIBRARY (nint) NWDPJobMove (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        destPrinterRef,
   pNWDPJobRef       jobRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

destPrinterRef

(IN) Specifies the reference to the destination printer.

jobRefPtr

(IN) Points to the reference of the moved job object.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

For the *jobRefPtr* parameter, the job reference is updated to reflect the new destination printer.

new destination printer.

The job identifier is different after the move to the new printer. If an application displays the job identifier, it should call the **NWDPJobGetId** function to refresh the job identifier.

If the source and destination printers are controlled-access printers (represented in NDS), they must reside within the same tree.

If the **NWDPJobMove** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00090005L NWDP_LE_JOB_NO_SUCH_JOB
0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x030A0001L NWDP_EC_NO_MEMORY
0x000C000DL NWDP_LE_PRT_DEST_TREE_DIFFERS

NCP Calls

None

See Also

NWDPJobCopy
NWDPPrtJobCopy
NWDPPrtJobMove

NWDPJobPause

Pauses the referenced job
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_job.h>

N_EXTERN_LIBRARY (nint) NWDPJobPause (
   NWDPAccessorRef   accessorRef,
   NWDPJobRef        jobRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

jobRef

(IN) Specifies the reference to the job object that is paused.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the job is already paused, N_SUCCESS is still returned.

If the **NWDPJobPause** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00090001L NWDP_LE_JOB_INVALID_OPERATION

0x00090005L NWDP_LE_JOB_NO_SUCH_JOB
0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN

NCP Calls

None

See Also

NWDPJobResume

NWDPJobPromote

Promotes the referenced job so it is next to be printer
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_job.h>

N_EXTERN_LIBRARY (nint) NWDPJobPromote (
   NWDPAccessorRef   accessorRef,
   NWDPJobRef        jobRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

jobRef

(IN) Specifies the reference to the job object that is promoted to the beginning of the job pool.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPJobPromote** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00090001L NWDP_LE_JOB_INVALID_OPERATION
0x00090005L NWDP_LE_JOB_NO_SUCH_JOB

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN

NCP Calls

None

See Also

NWDPJobReorder

NWDPJobReorder

Changes the position of the referenced job in the job pool
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_job.h>

N_EXTERN_LIBRARY (nint) NWDPJobReorder (
   NWDPAccessorRef   accessorRef,
   NWDPJobRef        jobRef,
   nuint32           referencedJobId);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

jobRef

(IN) Specifies the reference to the job object that is reordered.

referencedJobId

(IN) Specifies the job identifier that the job referenced by the *jobRef* parameter is placed behind.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the job object of the *referencedJobId* parameter is not found, the job object

If the job object of the *referencedJobId* parameter is not found, the job object referenced by the *jobRef* parameter is placed at the beginning of the job pool. Passing a zero (invalid job identifier) for the *referencedJobId* parameter has the same effect as calling the **NWDPJobPromote** function.

If the **NWDPJobReorder** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00090001L NWDP_LE_JOB_INVALID_OPERATION
0x00090005L NWDP_LE_JOB_NO_SUCH_JOB
0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN

NCP Calls

None

See Also

NWDPJobPromote

NWDPJobResume

Resumes a previously paused job
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_job.h>

N_EXTERN_LIBRARY (nint) NWDPJobResume (
   NWDPAccessorRef   accessorRef,
   NWDPJobRef        jobRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

jobRef

(IN) Specifies the reference to the job object that is resumed.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the job is already in a resumed state, N_SUCCESS is still returned.

If the **NWDPJobResume** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00090001L NWDP_LE_JOB_INVALID_OPERATION

0x00090005L NWDP_LE_JOB_NO_SUCH_JOB
0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN

NCP Calls

None

See Also

NWDPJobPause

NWDPJobSubmit

Closes the document data file and submits a job for printing
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_job.h>

N_EXTERN_LIBRARY (nint) NWDPJobSubmit (
   NWDPAccessorRef   accessorRef,
   pNWDPJobRef       jobRefPtr,
   nuint32           flags);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

jobRefPtr

(IN) Points to the reference of the job object in the MODB.

flags

(IN) Pass NWDP_JOB_FLG_DESTROY_REF to destroy the job reference after the job submission (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The **NWDPJobSubmit** function informs the printer that the job has the

The **NWDPJobSubmit** function informs the printer that the job has the attributes and the data requested by the application and that the job can be scheduled for printing. The job object must already exist in the MODB.

If the **NWDPJobSubmit** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00090001L NWDP_LE_JOB_INVALID_OPERATION
0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN

NCP Calls

None

See Also

NWDPDocTransferFile
NWDPDocWriteBuf
NWDPJobCreateRef

NWDPJobValidateRef

Validates a job reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_job.h>

N_EXTERN_LIBRARY (nint) NWDPJobValidateRef (
   NWDPAccessorRef   accessorRef,
   NWDPJobRef        jobRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

jobRef

(IN) Specifies the job reference to validate.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The **NWDPJobValidateRef** function verifies that the job reference is associated with the *accessorRef* parameter when the application has received it from an untrusted third party.

NCP Calls

Print Service Group

None

NWDPLibCalloc

Allocates memory and sets memory to zero
Local Servers: nonblocking
Remote Servers: N/A
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
Service: Distributed Print

Syntax

```
#include <nwdp_lib.h>

N_EXTERN_LIBRARY (NWDPInitCode) NWDPLibCalloc (
   NWDPAccessorRef   accessorRef,
   pnptr             resultMemPtr,
   nint              numElements,
   nint              szMem);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

resultMemPtr

(OUT) Points to a pointer which will receive the newly allocated memory's address.

numElements

(IN) Specifies a number of array elements.

szMem

(IN) Specifies the amount of memory needed for each array element.

Return Values

0x00000000	N_SUCCESS
0x030A0001	NWDP_EC_NO_MEMORY
0x030A0009	NWDP_EC_ARTIFICAL_MEM_LIMIT
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR

0xFFFFFFFF	N_FAILURE

Remarks

The **NWDPLibCalloc** function is not recommended except for applications which must use this library.

The **NWDPLibCalloc** function can block if the server has to wait for allocation.

Any memory allocated by the **NWDPLibCalloc** function is automatically freed when the **NWDPLibTerm** function is called.

The size of memory allocated is (numElement*szMem).

The **NWDPLibCalloc** function is equivalent to the **calloc** function from the C run-time library except for error handling.

The errors NWDP_EC_NO_MEMORY and NWDP_EC_ARTIFICAL_MEM_LIMIT both return the following information:

The *otherError* field of the *accessorRef* will contain the size of the request made of the system, e.g., szMem, and

The *otherError2* field of the *accessorRef* will contain the current memory allocation total made through NWDPLibxxxx entry points

NCP Calls

None

NWDPLibFree

Frees memory allocated by other library routines
Local Servers: nonblocking
Remote Servers: N/A
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_lib.h>

N_EXTERN_LIBRARY (NWDPInitCode) NWDPLibFree (
   NWDPAccessorRef   accessorRef,
   nptr              memPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

memPtr

(IN) Points to the memory to be freed.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The **NWDPLibFree** function is equivalent to the **free** function.

The **NWDPLibFree** function is not recommended except for applications which must use this library.

For NLMs, memory is not returned to the system but is kept in a pool for

small allocations. This is done for performance reasons. All of these items are freed, except when the **NWDPLibTerm** function is called.

NCP Calls

None

NWDPLibInit

Initializes the library by allocating space on a per task/thread-group basis to identify the accessor and maintain copies of global values

Local Servers: nonblocking

Remote Servers: N/A

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_lib.h>

N_EXTERN_LIBRARY (NWDPInitCode) NWDPLibInit (
   nuint              expectedMajorVersion,
   nuint              expectedMinorVersion,
   nuint              expectedBugfixRevision,
   nuint              expectedPrereleaseVersion,
   nuint              taskID,
   nuint              subTaskID,
   pNWDPAccessorRef   accessorRefPtr,
   pnuint             returnedMajorVersionPtr,
   pnuint             returnedMinorVersionPtr,
   pnuint             returnedBugfixRevisionPtr,
   pnuint             returnedPrereleaseVersionPtr);
```

Parameters

expectedMajorVersion

(IN) Specifies the NWDP_LIBRARY_MAJOR_VERSION constant.

expectedMinorVersion

(IN) Specifies the NWDP_LIBRARY_MINOR_VERSION constant.

expectedBugfixRevision

(IN) Specifies the NWDP_LIBRARY_BUGFIX_REVISION constant.

expectedPrereleaseVersion

(IN) Specifies the NWDP_LIBRARY_PREREL_VERSION constant.

taskID

(IN) Specifies an operating system unique identifier for the current task/thread-group process that needs its own instance of the data for this session.

subTaskID

(IN) Specifies a subordinate identifier to the *taskID* parameter.

accessorRefPtr

(OUT) Points to the NWDPAccessorRef containing the accessor to be initialized.

returnedMajorVersionPtr

(OUT) Points to the unsigned integer containing the major version of this library.

returnedMinorVersionPtr

(OUT) Points to the unsigned integer containing the resulting minor version of this library.

returnedBugfixRevisionPtr

(OUT) Points to the unsigned integer containing the resulting bug-fix revision of this library.

returnedPrereleaseVersionPtr

(OUT) Points to the unsigned integer containing the resulting prerelease version of this library.

Return Values

0	N_SUCCESS
2	NWDP_IC_WARN_REPEAT_ID
3	NWDP_IC_WARN_VER_OBSOLETE
-2	NWDP_IC_ERR_NO_MEMORY
-3	NWDP_IC_ERR_NO_ACCESSOR
-4	NWDP_IC_ERR_WRONG_MAJOR_VERSION
-5	NWDP_IC_ERR_OLD_MINOR_VERSION
-6	NWDP_IC_ERR_OLD_REVISION
-7	NWDP_IC_ERR_IS_PRERELEASE
-8	NWDP_IC_ERR_OLD_PRERELEASE
-9	NWDP_IC_ERR_NWCALLS_INIT
-10	NWDP_IC_ERR_UNICODE_INIT
-11	NWDP_IC_ERR_WINDOW_CREATE
-12	NWDP_IC_ERR_MISMATCHED_DLL
-13	NWDP_IC_ERR_MISMATCHED_SUBLIB (only DOS)
-14	NWDP_IC_ERR_RPC_WINSOCK_INIT (only Win3x, Win95)

Remarks

The **NWDPLibInit** function can block if the server has to wait for allocation.

The following is an explanation of what version is older:

The oldest prerelease version is 0xD001 (the first development prerelease)

After subsequent development prereleases, the next newest is 0xA001 (the first alpha release)

After subsequent Alpha releases, the next newest is 0xB001 (the first beta release). After the Beta releases comes the final prerelease value. For the *expectedMajorVersion*, *expectedMinorVersion*, and *expectedBugfixRevision* parameters, the value is 0, (indicating not a prerelease).

If the *expectedPrereleaseVersion* parameter is zero and the library is older than this, NWDP_IC_ERR_IS_PRERELEASE is returned.

If the *expectedMajorVersion* parameter matches that of the library, functionality is guaranteed to be supported. If the major versions do not match, certain entry points are no longer supported and NWDP_IC_ERR_WRONG_MAJOR_VERSION is returned.

If the *expectedMajorVersion* parameter matches, but the *expectedMinorVersion* parameter is less than that of the library, the expected library may contain newly obsoleted entry points. If the library contains obsoleted entry points, NWDP_IC_WARN_VER_OBSOLETE is returned. This error is non-fatal because the entry points continue to be supported with all libraries of the same major version. If the *expectedMinorVersion* parameter is greater than that of the Library, NWDP_IC_ERR_OLD_MINOR_VERSION is returned.

If tests on the *expectedMajorVersion* and *expectedMinorVersion* parameters succeed, but the *expectedBugfixRevision* parameter is greater than the current one, NWDP_IC_ERR_OLD_REVISION is returned.

If tests on the *expectedMajorVersion*, *expectedMinorVersion* and *expectedBugfixRevision* parameters all succeed, but the *expectedPrereleaseVersion* parameter is "newer" than the one in the library, NWDP_IC_ERR_OLD_PRERELEASE is returned.

The *subTaskID* parameter allows an application to create multiple accessor references per task/thread-group process. Repeat usage of a *taskID*/*subTaskID* pair parameter causes the same accessor reference to be returned and a usage count to be incremented. It also provokes a warning error code, NWDP_IC_WARN_REPEAT_ID. This implies that the **NWDPLibTerm** function must be called with the same *accessorRef* parameter a number of times equal to the number of times the **NWDPLibInit** function is called.

A macro called **NWDPLibInitMac** has been defined with explicit error handling so that only the address of the accessor reference to be returned needs to be provided and all other parameters can have defaults. This

may be used as a code example, but since it contains a reference to exit(), it should not be used in production code.

NCP Calls

None

NWDPLibMalloc

Allocates memory
Local Servers: nonblocking
Remote Servers: N/A
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_lib.h>

N_EXTERN_LIBRARY (NWDPInitCode) NWDPLibMalloc (
   NWDPAccessorRef   accessorRef,
   pnptr             resultMemPtr,
   nint              szMem);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

resultMemPtr

(OUT) Points to a pointer which will receive the newly allocated memory's address.

szMem

(IN) Specifies the size of memory to allocate.

Return Values

0x00000000	N_SUCCESS
0x030A0001	NWDP_EC_NO_MEMORY
0x030A0009	NWDP_EC_ARTIFICAL_MEM_LIMIT
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The **NWDPLibMalloc** function is not recommended except for applications which must use this library.

The **NWDPLibMalloc** function can block if the server has to wait for allocation.

Any memory allocated by the **NWDPLibMalloc** function is automatically freed when the **NWDPLibTerm** function is called.

The **NWDPLibMalloc** function is equivalent to the **malloc** function from the C run-time library except for error handling.

The errors NWDP_EC_NO_MEMORY and NWDP_EC_ARTIFICAL_MEM_LIMIT both return the following information:

The *otherError* field of the *accessorRef* will contain the size of the request made of the system, e.g., szMem, and

The *otherError2* field of the *accessorRef* will contain the current memory allocation total made through NWDPLibxxxx entry point

NCP Calls

None

NWDPLibQMalloc

Allocates memory
Local Servers: nonblocking
Remote Servers: N/A
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_lib.h>

N_EXTERN_LIBRARY (NWDPInitCode) NWDPLibQMalloc (
   NWDPAccessorRef   accessorRef,
   pnptr             resultMemPtr,
   nint              szMem);
```

Parameters

accessorRef

(IN) Point to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

resultMemPtr

(OUT) Points to a pointer which will receive the newly allocated memory's address.

szMem

(IN) Specifies the size of memory to allocate.

Return Values

0x00000000	N_SUCCESS
0x030A0001	NWDP_EC_NO_MEMORY
0x030A0009	NWDP_EC_ARTIFICAL_MEM_LIMIT
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The **NWDPLibQMalloc** function is identical to the **NWDPLibMalloc** function except for the N_PLAT_NLM case when the non-blocking **_qmalloc** function is called instead of the **malloc** function.

The **NWDPLibQMalloc** function is not recommended except for applications which must use this library.

Any memory allocated by the **NWDPLibQMalloc** function is automatically freed when the **NWDPLibTerm** function is called.

The errors NWDP_EC_NO_MEMORY and NWDP_EC_ARTIFICAL_MEM_LIMIT both return the following information:

The *otherError* field of the *accessorRef* will contain the size of the request made of the system, e.g., szMem, and

The *otherError2* field of the *accessorRef* will contain the current memory allocation total made through NWDPLibxxxx entry points

NCP Calls

None

NWDPLibRealloc

Reallocates memory
Local Servers: nonblocking
Remote Servers: N/A
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_lib.h>

N_EXTERN_LIBRARY (NWDPInitCode) NWDPLibRealloc (
   NWDPAccessorRef   accessorRef,
   pnptr             resultMemPtr,
   nptr              sourceMemPtr,
   nint              szMem);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

resultMemPtr

(OUT) Points to a pointer which will receive the newly allocated memory's address.

sourceMemPtr

(IN) Points to the original allocation of memory which is to be copied and freed (replaced by the newer allocation).

szMem

(IN) Specifies the size of memory to allocate.

Return Values

0x00000000	N_SUCCESS
0x030A0001	NWDP_EC_NO_MEMORY
0x030A0009	NWDP_EC_ARTIFICAL_MEM_LIMIT

0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The **NWDPLibRealloc** function is not recommended except for applications which must use this library.

The **NWDPLibRealloc** function can block if the server has to wait for allocation.

Any memory allocated by the **NWDPLibRealloc** function is automatically freed when the **NWDPLibTerm** function is called.

The **NWDPLibRealloc** function is equivalent to the **realloc** function from the C run-time library except for the error handling.

The errors NWDP_EC_NO_MEMORY and NWDP_EC_ARTIFICAL_MEM_LIMIT both return the following information:

The *otherError* field of the *accessorRef* will contain the size of the request made of the system, e.g., szMem, and

The *otherError2* field of the *accessorRef* will contain the current memory allocation total made through NWDPLibxxxx entry points

NCP Calls

None

NWDPLibSetArtificialMemLimit

Allows testing of error handling paths in the library user's code by forcing the nth **malloc** function to fail

Local Servers: nonblocking

Remote Servers: N/A

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_lib.h>

N_EXTERN_LIBRARY (nint) NWDPLibSetArtificialMemLimit (
   NWDPAccessorRef   accessorRef,
   nuint32           mallocsBeforeMallocFail);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

mallocsBeforeMallocFail

(IN) Specifies the number of calls to the **NWDPLibCalloc** function, **NWDPLibMalloc** function, **NWDPLibRealloc** function, and other uses internal to the library before a failure is to be generated.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the *mallocsBeforeMallocFail* parameter is set to NWDP_MEM_LIMIT_NONE (the initial value), the artificial memory

limit feature of the library is disabled.

NCP Calls

None

NWDPLibSetArtificialIODelay

Sets the number of milliseconds to delay all I/O requests if N_DEBUG is defined

Local Servers: nonblocking

Remote Servers: N/A

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_lib.h>

N_EXTERN_LIBRARY (nint) NWDPLibSetArtificialIODelay (
   NWDPAccessorRef   accessorRef,
   nuint32           milliseconds);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

milliseconds

(IN) Specifies the number of milliseconds to delay after each I/O function or request/reply function.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The delay is placed so that observed synchronization between client and server does not appear confused. For example, if you want to observe the change in a server while operating a client in the debugger, the I/O

function executes immediately after you step over the function.

Under DOS or Windows3.x, a calibration (requiring two seconds) is performed the first time the **NWDPLibSetArtificialIODelay** function is called with a non-zero value for milliseconds.

NCP Calls

None

NWDPLibTerm

Terminates the library's usage for the associated task and subtask IDs and decrements a usage counter

Local Servers: nonblocking

Remote Servers: N/A

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_lib.h>

N_EXTERN_LIBRARY (nint) NWDPLibTerm (
   pNWDPAccessorRef   accessorRefPtr);
```

Parameters

accessorRefPtr

(IN) Points to the accessor reference to be terminated.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the usage counter gets set to zero, the **NWDPLibTerm** function also deallocates all overhead space allocated to the accessor reference.

NCP Calls

None

See Also

Print Service Group

NWDPLibInit

NWDPNSrvAddBrokerObject

Creates a broker object with the specified name in the name service database and creates a reference to the object

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_nsr.h>

N_EXTERN_LIBRARY (nint) NWDPNSrvAddBrokerObject (
   NWDPAccessorRef   accessorRef,
   pnstr16           objectRDN16Ptr,
   pnstr16           resourceMgrVolObjectRDN16Ptr,
   pnstr16           resourceMgrDatabasePath16Ptr,
   pNWDPNSrvObjRef   nsObjRefPtr,
   nuint32           flags);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

objectRDN16Ptr

(IN) Points to the relative distinguished name (RDN) of the broker object to be created.

resourceMgrVolObjectRDN16Ptr

(IN) Points to the RDN of the proposed volume object to be used by the broker's resource manager.

resourceMgrDatabasePath16Ptr

(IN) Points to the proposed volume object to be used by the broker's resource manager.

nsObjRefPtr

(OUT) Points to the NWDPNSrvObjRef containing the resulting object in the name service (optional).

flags

(IN) Specify services to enable.

Return Values

0x00000000	N_SUCCESS

Remarks

For the *flags* parameter, the following may be ORed together;

0x00000001 NWDP_NSRV_FLG_ENABLE_REGIST_SVC
0x00000002 NWDP_NSRV_FLG_ENABLE_RESMAN_SVC
0x00000004 NWDP_NSRV_FLG_ENABLE_NOTIFY_SVC

If NWDP_NSRV_FLG_ENABLE_RESMAN_SVC is not present in the *flags* parameter, the *resourceMgrVolObjectRDN16Ptr* and *resourceMgrDatabasePathRDN16Ptr* parameters may be NULL.

NCP Calls

None

NWDPNSrvAddPrinterObject

Creates a printer object with the specified name in the name service database and creates a reference to the object

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_nsr.h>

N_EXTERN_LIBRARY (nint) NWDPNSrvAddPrinterObject (
   NWDPAccessorRef   accessorRef,
   pnstr16           objectRDN16Ptr,
   NWDPNSrvObjRef    nsObjRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

objectRDN16Ptr

(IN) Points to the name of the printer to be created.

nsObjRefPtr

(OUT) Points to the NWDPNSrvObjRef containing the resulting object in the name service (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPNSrvAddPrinterObject** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER
0x01000005L NWDP_EC_NDS
0x030A0001L NWDP_EC_NO_MEMORY
0x000B0001L NWDP_LE_NSRV_NO_CONTEXT_IN_USE
0x000B0002L NWDP_LE_NSRV_OBJECT_EXISTS

NCP Calls

None

NWDPNSrvAddPSMObject

Creates a print services manager object with the specified name in the name service database and creates a reference to the object

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_nsr.h>

N_EXTERN_LIBRARY (nint) NWDPNSrvAddPSMObject (
   NWDPAccessorRef   accessorRef,
   pnstr16           objectRDN16Ptr,
   pnstr16           databaseVolObjectRDN16Ptr,
   pNWDPNSrvObjRef   nsObjRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

objectRDN16Ptr

(IN) Points to the name of the print services manager to be created.

databaseVolObjectRDN16Ptr

(IN) Points to the volume object name where the print services manager stores data.

nsObjRefPtr

(OUT) Points to the NWDPNSrvObjRef containing the resulting object in the name service (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

FF	

Remarks

If the **NWDPNSrvAddPSMObject** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER
0x01000005L NWDP_EC_NDS
0x030A0001L NWDP_EC_NO_MEMORY
0x000B0001L NWDP_LE_NSRV_NO_CONTEXT_IN_USE
0x000B0002L NWDP_LE_NSRV_OBJECT_EXISTS

NCP Calls

None

NWDPNSrvCompareNDSObjectNames

Compares two object names for equality of the referenced objects
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_nsr.h>

N_EXTERN_LIBRARY (nint) NWDPNSrvCompareNDSObjectNames (
   NWDPAccessorRef   accessorRef,
   pnstr16           treeName16Ptr, /* may be NULL */
   pnstr16           object1Name16Ptr,
   pnstr16           object2Name16Ptr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

treeName16Ptr

(IN) Points to the Unicode tree name.

object1Name16Ptr

(IN) Points to the first Unicode object name.

object2Name16Ptr

(IN) Points to the second Unicode object name.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFF	N_FAILURE

Remarks

This comparison is not textual. It finds an NDS server within the specified tree and evaluates the two strings by mapping them to their respective NDS object Ids. If the Ids are identical, the return value is N_SUCCESS.

NCP Calls

None

NWDPNSrvCreateRef

Creates a reference to the object identified by the object name string and type

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_nsr.h>

N_EXTERN_LIBRARY (nint) NWDPNSrvCreateRef (
   NWDPAccessorRef   accessorRef,
   pnstr16           objectRDN16Ptr,
   NWDPNSrvObjType   objectType,
   pNWDPNSrvObjRef   nsObjRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

objectRDN16Ptr

(IN) Points to the name of the object in the name service.

objectType

(IN) Specifies the NWDPNSrvObjType structure containing an NWDP_NSRV_OT_ constant.

nsObjRefPtr

(OUT) Points to the NWDPNSrvObjRef containing the resulting object in the name service.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

FF	

Remarks

The **NWDPNSrvSetNativeNDSContext** function must be called before calling the **NWDPNSrvCreateRef** function.

The name service is accessed only to verify the existence of the object and its type.

For name services that require types to make the object name unique, a translation is provided internal to the library. For name services that do not have this feature, the constant is used in the debug version library to provide parameter checking in other calls.

The *objectType* parameter can have the following values:

-1 NWDP_NSRV_OT_ANY_MASK
1 NWDP_NSRV_OT_BROKER
2 NWDP_NSRV_OT_PRINTER
4 NWDP_NSRV_OT_MANAGER

If the **NWDPNSrvCreateRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER
0x01000002L NWDP_EC_PARAM_VAL_UNRECOGNIZED
0x01000005L NWDP_EC_NDS
0x030A0001L NWDP_EC_NO_MEMORY
0x000B0001L NWDP_LE_NSRV_NO_CONTEXT_IN_USE
0x000B0003L NWDP_LE_NSRV_NO_SUCH_OBJECT

NCP Calls

None

NWDPNSrvDestroyRef

Unconditionally releases resources allocated to the object reference and sets its value to zero

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_nsr.h>

N_EXTERN_LIBRARY (nint) NWDPNSrvDestroyRef (
   NWDPAccessorRef   accessorRef,
   pNWDPNSrvObjRef   nsObjRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

nsObjRefPtr

(IN) Points to the NWDPNSrvObjRef containing the object in the name service.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPNSrvDestroyRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER

0x000A0002L NWDP_LE_MEMORY_NOT_MALLOCED
0x000B0001L NWDP_LE_NSRV_NO_CONTEXT_IN_USE

NCP Calls

None

NWDPNSrvGetNativeNDSContext

Returns the native NDS context handle of the context currently in use and allows access to other features of NDS using the native NDS functions

Local Servers: nonblocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_nsr.h>

N_EXTERN_LIBRARY (nint) NWDPNSrvGetNativeNDSContext (
   NWDPAccessorRef   accessorRef,
   pnuint32          contextHandlePtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

contextHandlePtr

(OUT) Points to the resulting NDS context handle.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPNSrvGetNativeNDSContext** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000005L NWDP_EC_NDS

0x000B0001L NWDP_LE_NSRV_NO_CONTEXT_IN_USE

NCP Calls

None

NWDPNSrvListObjects

Returns through the **NWDPNSrvObjListCallback** function the name of each object of the specified type

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_nsr.h>

N_EXTERN_LIBRARY (nint) NWDPNSrvListObjects (
   NWDPAccessorRef           accessorRef,
   NWDPNSrvObjType          objectTypeFilter,
   NWDPNSrvObjListCallback   listCallBack,
   nparam                     callerDefinedParam);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

objectTypeFilter

(IN) Specifies the NWDPNSrvObjType structure containing an NWDP_NSRV_OT_ constant.

listCallBack

(IN) Specifies the **NWDPNSrvObjListCallback** function returning the results of the list command.

callerDefinedParam

(IN) Specifies a user-defined parameter passed to the **NWDPNSrvObjListCallback** function containing application-specific data.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR

0xFFFFFFFF	N_FAILURE

Remarks

The returned objects are listed in the name service database.

The *objectType* parameter can have the following values:

-1 NWDP_NSRV_OT_ANY_MASK
1 NWDP_NSRV_OT_BROKER
2 NWDP_NSRV_OT_PRINTER
4 NWDP_NSRV_OT_MANAGER

The *callerDefinedParam* parameter can contain the handle to the list box where the list is to be inserted.

If the **NWDPNSrvListObjects** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER
0x000A0003L NWDP_LE_CANCEL_LIST

NCP Calls

None

See Also

NWDPNSrvObjListCallback

NWDPNSrvMakeFQNFromObject

Creates a Fully Qualified Name (FQN) from an object name
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_nsr.h>

N_EXTERN_LIBRARY (nint) NWDPNSrvMakeFQNFromObject (
   NWDPAccessorRef      accessorRef,
   pnstr16              objectRDN16Ptr,
   nuint                sizeOfBuffer,
   pNWDPNSrvFQNBuffer   fqnBufferPtr,
   pnuint               sizeOfResultPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

objectRDN16Ptr

(IN) Points to the object name in the name service.

sizeOfBuffer

(IN) Specifies the length in bytes of the FQN buffer whose address is given by the *fqnBufferPtr* parameter.

fqnBufferPtr

(OUT) Points to the NWDPNSrvFQNBuffer structure receiving the FQN (must be MAX_DN_BYTES in size to allow for worst case).

sizeOfResultPtr

(OUT) Points to the number of bytes (i.e., strlen(fqnBufferPtr) plus one) consumed by the result (this is useful when the buffer size is insufficient).

Return Values

0x00000000	N_SUCCESS

0	
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPNSrvMakeFQNFromObject** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER
0x01000005L NWDP_EC_NDS
0x000A0006L NWDP_LE_RESULT_BUFFER_TOO_SMALL
0x000B0001L NWDP_LE_NSRV_NO_CONTEXT_IN_USE
0x000B0003L NWDP_LE_NSRV_NO_SUCH_OBJECT

NCP Calls

None

NWDPNSrvModifyAttrs

Modifies attributes contained in the referenced name service object
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_nsr.h>

N_EXTERN_LIBRARY (nint) NWDPNSrvModifyAttrs (
   NWDPAccessorRef          accessorRef,
   NWDPNSrvObjRef           nsObjRef,
   NWDPAttrSetRef           modifyRef,
   NWDPModifyOperatorEnum   nsrvOperator);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

nsObjRef

(IN) Specifies the NWDPNSrvObjRef containing the name service object.

modifyRef

(IN) Specifies the NWDPAttrSetRef containing a list of attribute value pairs used to modify the name service object.

nsrvOperator

(IN) Specifies the modify operation to be applied to all attributes in the set (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

FF	

Remarks

The *nsrvOperator* parameter can have the following values:

0 NWDP_MODIFY_OP_NULL
1 NWDP_MODIFY_OP_REPLACE
2 NWDP_MODIFY_OP_ADD_VALUES
3 NWDP_MODIFY_OP_REMOVE_VALUES
5 NWDP_MODIFY_OP_REMOVE_ATTRIBUTE

If the *nsrvOperator* parameter is not NWDP_MODIFY_OP_NULL, this value will be placed in the *qualifier* field of each NWDPAttribute structure in the set.

If the **NWDPNSrvModifyAttrs** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000005L NWDP_EC_NDS
0x030A0001L NWDP_EC_NO_MEMORY
0x000B0001L NWDP_LE_NSRV_NO_CONTEXT_IN_USE
0x000B0003L NWDP_LE_NSRV_NO_SUCH_OBJECT
0x000B0004L NWDP_LE_NSRV_WRONG_OBJECT_TYPE

NCP Calls

None

NWDPNSrvObjGetAttributeSet

Returns an attribute set for the specified object
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_nsr.h>

N_EXTERN_LIBRARY (nint) NWDPNSrvObjGetAttributeSet (
   NWDPAccessorRef   accessorRef,
   NWDPNSrvObjRef    nsObjRef,
   NWDPOidSetRef     requestedOidSetRef,
   pNWDPAttrSetRef   resultAttrSetRefPtr,
   pNWDPASAVPRef     resultAvpRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

nsObjRef

(IN) Specifies the NWDPNSrvObjRef containing the name service object.

requestedOidSetRef

(IN) Specifies the NWDPOidSetRef containing a list of attributes whose values are to be returned.

resultAttrSetRefPtr

(IN) Points to the NWDPAttrSetRef containing the resulting attribute set.

resultAvpRefPtr

(OUT) Points to the NWDPASAVPRef (optional).

Return Values

0x00000000	N_SUCCESS

0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPNSrvObjGetAttributeSet** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000005L NWDP_EC_NDS
0x030A0001L NWDP_EC_NO_MEMORY
0x000B0001L NWDP_LE_NSRV_NO_CONTEXT_IN_USE
0x000B0003L NWDP_LE_NSRV_NO_SUCH_OBJECT
0x000B0004L NWDP_LE_NSRV_WRONG_OBJECT_TYPE

NCP Calls

None

NWDPNSrvObjListCallback

Returns through the **NWDPNSrvListObjects** function the name of each object of the specified type

Local Servers: N/A

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_nsr.h>

N_TYPEDEF_CALLBACK (nint, NWDPNSrvObjListCallback) (
   NWDPAccessorRef        accessorRef,
   nparam                 listCallbackParam2,
   nint                   totalCallsToBeMade,
   nint                   currentCallCount,
   pNWDPNSrvObjListItem   itemReceivedPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

listCallbackParam2

(IN) Specifies a user-defined contextual variable.

totalCallsToBeMade

(IN) Specifies the number of calls to be made to the **NWDPNSrvObjListCallback** function (equals the number of objects plus one).

currentCallCount

(IN) Specifies the current call number.

itemReceivedPtr

(IN) Points to the NWDPNSrvObjListItem structure (NULL is passed back on the last callback).

Return Values

0x00000000	N_SUCCESS

0	
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

NCP Calls

None

See Also

NWDPNSrvListObjects

NWDPNSrvQueryRoleMembership

Returns rights to the specified object as role membership boolean values
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_nsr.h>

N_EXTERN_LIBRARY (nint) NWDPNSrvQueryRoleMembership (
   NWDPAccessorRef   accessorRef,
   NWDPNSrvObjRef    nsObjRef,
   pnstr16           memberRDN16Ptr,
   pnbool            isAManagerPtr,
   pnbool            isAnOperatorPtr,
   pnbool            isAUserPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

nsObjRef

(IN) Specifies a reference to the printer, manager, or broker object.

memberRDN16Ptr

(IN) Points to a user relative distinguished object name in Unicode (optional).

isAManagerPtr

(OUT) Points to the boolean indicating the member named above having the role of manager (optional).

isAnOperatorPtr

(OUT) Points to the boolean indicating the member named above having the role of operator (optional).

isAUserPtr

(OUT) Points to the boolean indicating the member named above having the role of user (optional).

Return Values

0x00000000	N_SUCCESS

Remarks

If the *memberRDN16Ptr* parameter is NULL, the **NWDPNSrvQueryRoleMembership** function calls NWDSWhoAmI in order to provide an implicit name for the caller.

Although the *isAManagerPtr*, *isAnOperatorPtr*, and *isAUserPtr* parameters are all optional, at least one of them must be supplied.

NCP Calls

None

NWDPNSrvRemoveObject

Removes the object from the name service database and destroys the reference to the passed object

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_nsr.h>

N_EXTERN_LIBRARY (nint) NWDPNSrvRemoveObject (
   NWDPAccessorRef   accessorRef,
   pNWDPNSrvObjRef   nsObjRefPtr,
   nuint32           flags);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

nsObjRefPtr

(IN) Points to the NWDPNSrvObjRef containing the object in the name service.

flags

(IN) Specify options for a remove.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the *flags* parameter contains the NWDP_NSRV_FLG_DONT_DEL_OBJECT bit, clean up of attributes, and so forth, related to the object will be done, but the **NWDSRemoveObject** function must be called by the user.

If the **NWDPNSrvRemoveObject** function is successful, the NWDPNSrvObjRef is set to zero.

If the **NWDPNSrvRemoveObject** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER
0x01000005L NWDP_EC_NDS
0x000B0003L NWDP_LE_NSRV_NO_SUCH_OBJECT

NCP Calls

None

NWDPNSrvSetNativeNDSContext

Sets the library's implicit name service to NetWare Directory Services (NDS) and defines the context variable to be used in all implicit references to name service objects

Local Servers: nonblocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_nsr.h>

N_EXTERN_LIBRARY (nint) NWDPNSrvSetNativeNDSContext (
   NWDPAccessorRef   accessorRef,
   nuint32           contextHandle);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

contextHandle

(IN) Specifies the NDS context handle created by calling the **NWDSCreateContext** function.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

You must be logged in and authenticated to call the **NWDPNSrvSetNativeNDSContext** function.

NOTE: Take care when altering the contents of the NDS context after calling the **NWDPNSrvSetNativeNDSContext** function because all Distributed Print Services functions depend on this same value. No information other than the value of the *contextHandle* parameter is preserved in the structure referenced by the *accessorRef* parameter.

If the **NWDPNSrvSetNativeNDSContext** function returns N_FAILURE, the NWPDLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER
0x01000005L NWDP_EC_NDS

NCP Calls

None

NWDPNSrvValidateObjectRef

Validates the origin of the NWDPNSrvObjRef
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_nsr.h>

N_EXTERN_LIBRARY (nint) NWDPNSrvValidateObjectRef (
   NWDPAccessorRef   accessorRef,
   NWDPNSrvObjRef    objectRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

objectRef

(IN) Specifies an NWDPNSrvObjRef.

Return Values

0x00000000	N_SUCCESS

Remarks

The **NWDPSrvValidateObjectRef** function verifies that the object reference is associated with the *accessorRef* parameter when the application has received it from an untrusted third party.

NCP Calls

None

NWDPNtfyAddDetailEventObject

Adds a detail event object (set of specified events) to a profile
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyAddDetailEventObject (
   NWDPAccessorRef             accessorRef,
   NWDPNtfyProfileRef          profileRef,
   nuint32                     eventType,
   pNWDPOid                    containingClassOidPtr,
   pNWDPObjectIdentification   containingObjectIdOptionPtr,
   NWDPOidSetRef                eventOidSetRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

profileRef

(IN) Specifies the reference to the profile where the detail event object is added.

eventType

(IN) Specifies the event type of the new event object.

containingClassOidPtr

(IN) Points to the containing object class: printer or job.

containingObjectIdOptionPtr

(IN) Points to the object whose events are to be monitored for notification (if NULL is passed, the library will place the printer agent name in this field).

eventOidSetRef

(IN) Specifies the specific events to register for notification.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

A detail event object is an event object containing either printer or job specific events which are represented by OIDs. When one of the printer or job specific events occur, notification is sent.

The *profileRef* parameter points to the NWDPEventHandlingProfile structure. The **NWDPNtfyAddEvent** function only changes the information contained in this structure.To register this profile with the notification service, call the **NWDPNtfyAddProfile** function or the **NWDPNtfyModifyProfile** function for existing profiles.

The *eventType* parameter can be the following abstract event types:

NWDP_EVENT_TYPE_ERROR
NWDP_EVENT_TYPE_REPORT
NWDP_EVENT_TYPE_WARNING

The other event type is NWDP_EVENT_TYPE_VALUE_CHANGE.

The OID identified by the *containingClassOidPtr* parameter is the same OID returned in the NWDPEventObjectId structure from the **NWDPNtfyGetSupportedEvents** function.

For the *containingObjectIdOptionPtr* parameter, when this profile is submitted with a job attribute, set the designator to NWDP_OBJ_ID_PRT_CONTAIND_OBJ_ID. Leave the remainder of the NWDPObjectIdentification structure initialized to zero.

For the *eventOidSetRef* parameter, each event is represented by an OID. To obtain a list of supported events, call the **NWDPNtfyGetSupportedEvents** function.

If the **NWDPNtfyAddDetailEventObject** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPNtfyAddProfile
NWDPNtfyCreateProfileRef
NWDPNtfyGetSupportedEvents
NWDPNtfyModifyProfile
NWDPOSCreateRef

NWDPNtfyAddEvent

Adds an event to a detail event object
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyAddEvent (
   NWDPAccessorRef             accessorRef,
   NWDPNtfyProfileRef          profileRef,
   nuint32                     eventType,
   pNWDPOid                    containingClassOidPtr,
   pNWDPObjectIdentification   containingObjectIdOptionPtr,
   pNWDPOid                     eventOidPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

profileRef

(IN) Specifies the reference to the profile containing the detail event object (the event is added to this event object).

eventType

(IN) Specifies the event type of the detail event object.

containingClassOidPtr

(IN) Points to the containing object class: printer or job.

containingObjectIdOptionPtr

(IN) Points to the object whose events are monitored for notification (if NULL is passed, the library places the printer agent name in this field).

eventOidPtr

(IN) Points to the added event.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

A detail event object is an event object containing either printer or job specific events, which are represented by OIDs. When one of the printer or job specific events occur, notification is sent.

The *profileRef* parameter points to the NWDPEventHandlingProfile structure. The **NWDPNtfyAddEvent** function only changes the information contained in this structure. To register this profile with the notification service, call the **NWDPNtfyAddProfile** function or the **NWDPNtfyModifyProfile** function for existing profiles.

The *eventType* parameter can be the following abstract event types:

NWDP_EVENT_TYPE_ERROR
NWDP_EVENT_TYPE_REPORT
NWDP_EVENT_TYPE_WARNING

The other event type is NWDP_EVENT_TYPE_VALUE_CHANGE.

The *eventType*, *containingClassOidPtr*, and *containingObjectIdOptionPtr* parameters determine which existing event object receives the new event.

For the *containingClassOidPtr* parameter, the same class is returned in the NWDPEventObjectId structure when you call the NWDPNtfyGetSupportedEvents function.

For the *eventOidPtr* parameter, call the **NWDPNtfyGetSupportedEvents** function to obtain a list of supported events.

If the **NWDPNtfyAddEvent** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00120002L NWDP_LE_NTFY_EVENT_OBJ_NOT_FND
0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

Print Service Group

NWDPNtfyAddProfile
NWDPNtfyGetSupportedEvents
NWDPNtfyModifyProfile
NWDPNtfyRemoveEvent

NWDPNtfyAddFilterEventObject

Adds a filter event object to a profile
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyAddFilterEventObject(
   NWDPAccessorRef             accessorRef,
   NWDPNtfyProfileRef          profileRef,
   nuint32                     eventType,
   pNWDPOid                    containingClassOidPtr,
   pNWDPObjectIdentification   containingObjectIdOptionPtr,
   pNWDPOid                  filterClassOidPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

profileRef

(IN) Specifies the notification service reference where the filter event is added.

eventType

(IN) Specifies the event type of the new event object.

containingClassOidPtr

(IN) Points to the containing object class: printer or job.

containingObjectIdOptionPtr

(IN) Points to the object which has events monitored for notification (if NULL is passed, the library places the printer agent name in this field).

filterClassOidPtr

(IN) Points to the filter that filters the selection of events registered for notification on the following classes: Media Path, Marker, Input, and so forth.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *profileRef* parameter points to the NWDPEventHandlingProfile structure. The **NWDPNtfyAddFilterEventObject** function only changes the information contained in this structure. To register this profile with the notification service, call the **NWDPNtfyAddProfile** function or the **NWDPNtfyModifyProfile** function for existing profiles.

The *eventType* parameter can be the following abstract event types:

NWDP_EVENT_TYPE_ERROR
NWDP_EVENT_TYPE_REPORT
NWDP_EVENT_TYPE_WARNING

The other event type is NWDP_EVENT_TYPE_VALUE_CHANGE.

The OID identified by the *containingClassOidPtr* parameter is the same OID returned in the NWDPEventObjectId structure from the **NWDPNtfyGetSupportedEvents** function.

When the profile of the *containingObjectIdOptionPtr* parameter is being submitted with a job attribute, set the designator to NWDP_OBJ_ID_PRT_CONTAIND_OBJ_ID. Leave the remainder of the NWDPObjectIdentifier structure initialized at zero.

The following items cause notification to be sent of all error events occurring on the printer's input object:

a filter event object with a filter of Input

a containing class of Printer

an abstract event type of NWDP_EVENT_TYPE_ERROR

If the **NWDPNtfyAddFilterEventObject** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPNtfyAddProfile
NWDPNtfyCreateProfileRef
NWDPNtfyModifyProfile

NWDPNtfyAddMethod

Adds a notification method to the notification service
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyAddMethod (
   NWDPAccessorRef   accessorRef,
   NWDPNtfyRef       notifyRef,
   pnstr16           fileName16Ptr,
   pNWDPNameOrOid    methodIdPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

notifyRef

(IN) Specifies the notification service reference with the added method.

fileName16Ptr

(IN) Points to the NLM name loaded on the server (this is a Unicode string).

methodIdPtr

(OUT) Points to the method identifier that is added (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the *methodIdPtr* parameter is non-NULL, the **NWDPAUtilFreeDataType** function must be called to free allocated memory (use the NWDP_AVT_NAME_OR_OID for the *dataType* parameter).

If the **NWDPNtfyAddMethod** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x03060003L NWDP_EC_NOTIFY_RESULT
0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPNtfyRemoveMethod

NWDPNtfyAddObjectEventObject

Adds an event object of the type object to a profile
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyAddObjectEventObject (
   NWDPAccessorRef             accessorRef,
   NWDPNtfyProfileRef          profileRef,
   nuint32                     eventType,
   pNWDPOid                    containingClassOidPtr,
   pNWDPObjectIdentification   containingObjectIdOptionPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

profileRef

(IN) Specifies the profile reference where the event object of the object is added.

eventType

(IN) Specifies the event type of the new event object.

containingClassOidPtr

(IN) Points to the containing object class: printer or job.

containingObjectIdOptionPtr

(IN) Points to the object which has events monitored for notification (if NULL is passed, the library places the printer agent name in this field).

Return Values

0x00000000	N_SUCCESS

0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *profileRef* parameter points to the NWDPEventHandlingProfile structure. The **NWDPNtfyAddObjectEventObject** function only changes the information contained in this structure. To register this profile with the notification service, call the **NWDPNtfyAddProfile** function or the **NWDPNtfyModifyProfile** function for the existing profiles.

The *eventType* parameter can be the following abstract event types:

NWDP_EVENT_TYPE_ERROR
NWDP_EVENT_TYPE_REPORT
NWDP_EVENT_TYPE_WARNING

The other event type is NWDP_EVENT_TYPE_VALUE_CHANGE.

If an *eventType* parameter is set to NWDP_EVENT_TYPE_ERROR and the *containingClassOidPtr* parameter is in the printer class, the notification for all printer errors is registered.

The OID identified by the *containingObjectIdOptionPtr* parameter is the same OID returned in the NWDPEventObjectId structure from the **NWDPNtfyGetSupportedEvents** function.

When this profile is submitted with a job as one of the attributes, set the designator to NWDP_OBJ_ID_PRT_CONTAIND_OBJ_ID for the *containingIdObjectOptionPtr* parameter. The remainder of the NWDPObjectIdentification structure is initialized at zero.

If the **NWDPNtfyAddObjectEventObject** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* parameter contains the following:

0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPNtfyAddProfile
NWDPNtfyCreateProfileRef
NWDPNtfyModifyProfile

NWDPNtfyAddProfile

Adds a new profile to the notification service
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyAddProfile (
   NWDPAccessorRef               accessorRef,
   NWDPNtfyProfileRef            profileRef,
   NWDPNtfyProgrammaticCallback   callBack);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

profileRef

(IN) Specifies the reference of the profile added to the notification service.

callBack

(IN) Specifies the application-supplied callback function called by the service each time a notification is set (pass NULL, unless the*methodId* field is set to programmatic event notification).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

Remarks

For persistent profiles, the *profileRef* parameter identifies which NDS printer object stores the profile.

The *profileRef* parameter points to the NWDPEventHandlingProfile structure. The profile information in this structure is registered with the notification service.

If the **NWDPNtfyAddProfile** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* parameter contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x01000001L NWDP_EC_INVALID_PARAMETER

NCP Calls

None

See Also

NWDPNtfyAddDetailEventObject
NWDPNtfyAddFilterEventObject
NWDPNtfyAddObjectEventObject
NWDPNtfyCreateProfileRef
NWDPNtfyRemoveProfile

NWDPNtfyCreateMethodInfoRef

Creates a method information reference that is used to retrieve information about a method

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyCreateMethodInfoRef (
   NWDPAccessorRef          accessorRef,
   NWDPNtfyRef              notifyRef,
   pNWDPNameOrOid           methodIdPtr,
   NWDPLanguageID           languageId,
   pNWDPNtfyMethodInfoRef   methodInfoRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

notifyRef

(IN) Specifies the notification service reference.

methodIdPtr

(IN) Points to the method from which information is retrieved.

languageId

(IN) Specifies the language of the returned information (pass NWDP_LID_DEFAULT to use the language identified by the environment variable NWLANGUAGE).

methodInfoRefPtr

(OUT) Points to the NWDPNtfyMethodInfoRef, the resulting method information reference.

Return Values

0x00000000	N_SUCCESS

0	
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPNtfyCreateMethodInfoRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x03060003L NWDP_EC_NOTIFY_RESULT
0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPNtfyDestroyMethodInfoRef

NWDPNtfyCreateProfileRef

Creates a profile reference based on caller-provided information
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyCreateProfileRef (
   NWDPAccessorRef         accessorRef,
   NWDPPrtRef              printerRef,
   pNWDPNtfyProfileInfo    profileInfoPtr,
   pNWDPNtfyProfileRef     profileRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the printer associated with the profile.

profileInfoPtr

(IN) Points to the NWDPNtfyProfileInfo structure containing the profile information.

profileRefPtr

(OUT) Points to the NWDPNtfyProfileRef, the resulting profile reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If persistence is set to NWDP_PERSISTENCE_PERMANENT in the *profileInfoPtr* parameter, the printer reference must be a NWDP_PRT_REF_TYPE type. However, if a named printer is installed, it could be a NWDP_PRT_REF_TYPE_INSTALLED type.

If the **NWDPNtfyCreateProfileRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x030A0001L NWDP_EC_NO_MEMORY
0x000A0008L NWDP_LE_LEF_NOT_CREATED_WITH_FQN_

NCP Calls

None

See Also

NWDPNtfyAddDetailEventObject
NWDPNtfyAddFilterEventObject
NWDPNtfyAddObjectEventObject
NWDPNtfyAddProfile
NWDPNtfyListPrompts
NWDPNtfyReleaseProfileRef

NWDPNtfyCreateProfileRefBOP

Creates a profile reference based on an existing profile
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyCreateProfileRefBOP (
   NWDPAccessorRef       accessorRef,
   NWDPPrtRef            printerRef,
   nuint32               profileId,
   pNWDPNtfyProfileRef   profileRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the printer associated with the profile.

profileId

(IN) Specifies the existing profile from which information is retrieved to create the reference.

profileRefPtr

(OUT) Points to the NWDPNtfyProfileRef, the resulting profile reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPNtfyCreateProfileRefBOP** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00120004L NWDP_LE_NTFY_NO_SUCH_PROFILE
0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPNtfyReleaseProfileRef

NWDPNtfyCreateRefBasedOnFQN

Creates a notification service reference based on the Fully Qualified Name (FQN) in DS

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyCreateRefBasedOnFQN (
   NWDPAccessorRef   accessorRef,
   pNWDPNSrvFQN      brokerFqnPtr,
   pNWDPNtfyRef      notifyRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

brokerFqnPtr

(IN) Points to the *NWDPNSrvFQN* structure containing the broker object distinguished name.

notifyRefPtr

(OUT) Points to the NWDPNtfyRef, the resulting notification reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

Remarks

If the **NWDPNtfyCreateRefBasedOnFQN** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* parameter contains the following:

0x01000005L NWDP_EC_NDS
0x030A0001L NWDP_EC_NO_MEMORY
0x000A000BL NWDP_LE_FQN_NOT_NDPS_BROKER

NCP Calls

None

See Also

NWDPNtfyDestroyRef

NWDPNtfyDestroyMethodInfoRef

Destroys a method information reference and frees all associated memory
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyDestroyMethodInfoRef (
   NWDPAccessorRef          accessorRef,
   pNWDPNtfyMethodInfoRef   methodInfoRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

methodInfoRefPtr

(IN) Points to the method information reference that is destroyed.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPNtfyDestroyMethodInfoRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* parameter contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER

NCP Calls

None

See Also

NWDPNtfyCreateMethodInfoRef

NWDPNtfyDestroyRef

Destroys a notification reference and frees all associated memory
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyDestroyRef (
   NWDPAccessorRef   accessorRef,
   pNWDPNtfyRef      notifyRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

notifyRefPtr

(IN) Points to the notification reference that is destroyed.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPNtfyDestroyRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* parameter contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER

NCP Calls

None

See Also

NWDPNtfyCreateRefBasedOnFQN

NWDPNtfyEventObjectListCallback

Lists event objects
Local Servers: N/A
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_TYPEDEF_CALLBACK (nint, NWDPNtfyEventObjectListCallback) (
   NWDPAccessorRef                accessorRef,
   nparam                         callerDefinedParam,
   nint                           totalCallsToBeMade,
   nint                           currentCallCount,
   pNWDPNtfyEventObjectListItem   itemReceivedPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

callerDefinedParam

(IN) Specifies a caller-defined contextual variable used by the callback function (optional).

totalCallsToBeMade

(IN) Specifies the number of times the **NWDPNtfyEventObjectListCallback** function is called (the number of event objects plus one).

currentCallCount

(IN) Specifies the current call number.

itemReceivedPtr

(IN) Points to the NWDPNtfyEventObjectListItem structure containing the event object information.

Return Values

0x00000000	N_SUCCESS

0	
0xFFFFFFFF	N_FAILURE

Remarks

For the *callerDefinedParam* parameter, this function uses the contextual variable to create the following:

a linked list head into which items can be added

a window handle where items can be displayed

any other data or combination of these two steps needed by the callback routine (optional)

For the *totalCallsToBeMade* parameter, the *getSupportedListItemPtr* parameter is NULL for the last callback.

NCP Calls

None

See Also

NWDPNtfyListEventObjects

NWDPNtfyGetDeliveryMethodInfo

Obtains the delivery information for a method
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyGetDeliveryMethodInfo (
   NWDPAccessorRef           accessorRef,
   NWDPNtfyMethodInfoRef     methodInfoRef,
   pNWDPNtfyDeliveryMethod  *deliveryMethodPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

methodInfoRef

(IN) Specifies the method reference needed to retrieve the information.

deliveryMethodPtr

(OUT) Points to the pointer of the NWDPNtfyDeliveryMethod structure (this pointer is assigned to the information for this method, thus, no memory is allocated).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The information returned is identical to the information returned in the **NWDPNtfyListMethods** function.

NCP Calls

None

See Also

NWDPNtfyCreateMethodInfoRef
NWDPNtfyListMethods

NWDPNtfyGetProfileId

Returns the registered profile identifier
Local Servers: nonblocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyGetProfileId (
   NWDPAccessorRef      accessorRef,
   NWDPNtfyProfileRef   profileRef,
   pnuint32             profileIdPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

profileRef

(IN) Specifies the registered profile reference.

profileIdPtr

(OUT) Points to the resulting identifier (pass the address of a nuint32).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPNtfyGetProfileId** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

NWDPLibErrorMac *(accessorRef)* contains the following:

0x00120003L NWDP_LE_NTFY_INVALID_PROFILE_ID

NCP Calls

None

See Also

NWDPNtfyAddProfile

NWDPNtfyGetProfileInfo

Retrieves profile information
Local Servers: nonblocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyGetProfileInfo (
   NWDPAccessorRef         accessorRef,
   NWDPNtfyProfileRef      profileRef,
   pNWDPNtfyProfileInfo    profileInfoPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

profileRef

(IN) Specifies the profile reference.

profileInfoPtr

(OUT) Points to the NWDPNtfyProfileInfo structure.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The information returned in the *profileInfoPtr* parameter is valid as long as the profile reference is not released.

the profile reference is not released.

NCP Calls

None

See Also

NWDPNtfyCreateProfileRef

NWDPNtfyGetSupportedEvents

Returns all supported events for the printer
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyGetSupportedEvents (
   NWDPAccessorRef                    accessorRef,
   NWDPPrtRef                         printerRef,
   NWDPNtfyGetSupportedListCallback   listCallBack,
   nparam                              callerDefinedParam);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the printer for which the supported events are returned.

listCallBack

(IN) Specifies the application-supplied callback function which is called by the **NWDPNtfyGetSupportedEvents** function for each abstract event object.

callerDefinedParam

(IN) Specifies a caller-defined parameter, containing application-specific data, that is passed to the callback function (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFF	N_FAILURE

0xFFFFFFFF	N_FAILURE

Remarks

Currently, there are six event objects for a given printer agent: Job Error, Job Warnings, Job Reports, Printer Errors, Printer Warnings, and Printer Reports. Each of these event objects has an associated attribute set. The attribute identifiers in these sets represent the support events. The values are the NWDP_AVT_OBJECT_IDENTIFIER type and identify the object class where the events occur. Examples of some values include: Printer, Marker, Input, Job, and Media Path. The values are used as conveniences for sorting.

If the **NWDPNtfyGetSupportedEvents** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x030A0001L NWDP_EC_NO_MEMORY
0x000A0005L NWDP_LE_UNKNOWN_CALLBACK_ERR

NCP Calls

None

NWDPNtfyGetSupportedListCallback

Lists the supported events
Local Servers: N/A
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_TYPEDEF_CALLBACK (nint, NWDPNtfyGetSupportedListCallback) (
   NWDPAccessorRef                accessorRef,
   nparam                         callerDefinedParam,
   nint                           totalCallsToBeMade,
   nint                           currentCallCount,
   pNWDPNtfyGetSupportedListItem   getSupportedListItemPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

callerDefinedParam

(IN) Specifies a caller-defined contextual variable used by the callback function (optional).

totalCallsToBeMade

(IN) Specifies the number of times the **NWDPNtfyGetSupportedListCallback** function is called (the number of event objects plus one).

currentCallCount

(IN) Specifies the current call number.

getSupportedListItemPtr

(IN) Points to the NWDPNtfyGetSupportedListItem structure containing the supported events.

Return Values

0x00000000	N_SUCCESS

0	
0xFFFFFFFF	N_FAILURE

Remarks

For the *callerDefinedParam* parameter, this function uses the contextual variable to create the following:

a linked list head into which items can be added

a window handle where items can be displayed

any other data or combination of these two steps needed by the callback routine (optional)

For the *totalCallsToBeMade* parameter, the *getSupportedListItemPtr* parameter is NULL for the last callback.

If the event attribute set reference needs to exist outside of the callback function, the NWDPASUseRef increments the usage count. The user is then responsible for calling NWDPASReleaseRef.

NCP Calls

None

See Also

NWDPASListAttributes
NWDPASListAttrValues
NWDPNtfyGetSupportedEvents

NWDPNtfyListEventObjects

Lists event objects for a profile
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyListEventObjects (
   NWDPAccessorRef                  accessorRef,
   NWDPNtfyProfileRef               profileRef,
   NWDPNtfyEventObjectListCallback  listCallback,
   nparam                            callerDefinedParam);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

profileRef

(IN) Specifies the profile reference containing event objects.

listCallback

(IN) Specifies the application-supplied callback function which is called by the **NWDPNtfyListEventObjects** function for each event object.

callerDefinedParam

(IN) Specifies a caller-defined parameter, containing application-specific data, that is passed to the callback function (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFF	N_FAILURE

0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPNtfyListEventObjects** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x000A0005L NWDP_LE_UNKNOWN_CALLBACK_ERR

NCP Calls

None

See Also

NWDPNtfyAddDetailEventObject
NWDPNtfyAddFilterEventObject
NWDPNtfyAddObjectEventObject
NWDPNtfyCreateProfileRef
NWDPNtfyCreateProfileRefBOP

NWDPNtfyListMethods

Lists the notification methods known to the notification service
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyListMethods (
   NWDPAccessorRef              accessorRef,
   NWDPNtfyRef                  notifyRef,
   NWDPLanguageID               languageId,
   NWDPNtfyMethodListCallback   listCallBack,
   nparam                        callerDefinedParam);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

notifyRef

(IN) Specifies the reference to the notification service.

languageId

(IN) Specifies the language used to list the methods (pass NWDP_LID_DEFAULT to use the language identified by the NWLANGUAGE environment variable).

listCallBack

(IN) Specifies the application-supplied callback function which is called by the **NWDPNtfyListMethods** function for each method.

callerDefinedParam

(IN) Specifies a caller-defined parameter, containing application-specific data, that is passed to the callback function (optional).

Return Values

0x0000000	N_SUCCESS

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPNtfyListMethods** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x03060003L NWDP_EC_NOTIFY_RESULT
0x000A0005L NWDP_LE_UNKNOWN_CALLBACK_ERR

NCP Calls

None

See Also

NWDPNtfyCreateRefBasedOnFQN

NWDPNtfyListProfiles

Lists the notification profiles for a printer
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyListProfiles (
   NWDPAccessorRef               accessorRef,
   NWDPPrtRef                    printerRef,
   pNWDPListProfilesChoice       profileChoicePtr,
   NWDPNtfyProfileListCallback   listCallback,
   nparam                         callerDefinedParam);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the printer reference.

profileChoicePtr

(IN) Points to the NWDPListProfilesChoice structure identifying the returned profiles (pass NULL to list all the profiles for the printer).

listCallback

(IN) Specifies the application-supplied callback function which is called by the **NWDPNtfyListProfiles** function for each profile.

callerDefinedParam

(IN) Specifies a caller-defined parameter, containing application-specific data, that is passed to the callback function (optional).

Return Values

0x00000000	N_SUCCESS

0	
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *profileChoicePtr* parameter can be set to list only specific profile identifiers using the *profileIdSeq* member of the NWDPListProfilesChoice structure. It can also be set to list profiles using the*filter* member, which consists of any combination of the following constraints:

consumerNameOption
methodIdOptionPtr
languageId

If the **NWDPNtfyListProfiles** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x030A0001L NWDP_EC_NO_MEMORY
0x000A0005L NWDP_LE_UNKNOWN_CALLBACK_ERR

NCP Calls

None

See Also

NWDPNtfyAddProfile

NWDPNtfyListPrompts

Lists the prompts associated with a notification method
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyListPrompts (
   NWDPAccessorRef              accessorRef,
   NWDPNtfyMethodInfoRef        methodInfoRef,
   NWDPNtfyPromptListCallback   listCallback,
   nparam                       callerDefinedParam);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

methodInfoRef

(IN) Specifies the delivery method for which the prompts are returned.

listCallback

(IN) Specifies the application-supplied callback function which is called by the **NWDPNtfyListPrompts** function for each prompt.

callerDefinedParam

(IN) Specifies a caller-defined parameter, containing application-specific data, that is passed to the callback function (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

FF	

Remarks

A User/Group/OU Name prompt might be returned for the PopUp delivery method. This information is used to receive the IE: user name input, which can be used to fill the NWDPNotifyDeliveryAddr structure. Notification delivery addresses (which specify where to send the notification) are needed for submitting notification profiles.

NCP Calls

None

See Also

NWDPNtfyCreateMethodInfoRef
NWDPNtfyCreateProfileRef

NWDPNtfyMethodListCallback

Lists methods
Local Servers: N/A
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_TYPEDEF_CALLBACK (nint, NWDPNtfyMethodListCallback) (
   NWDPAccessorRef           accessorRef,
   nparam                    callerDefinedParam,
   nint                      totalCallsToBeMade,
   nint                      currentCallCount,
   pNWDPNtfyDeliveryMethod   itemReceivedPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

callerDefinedParam

(IN) Specifies a caller-defined contextual variable used by the callback function (optional).

totalCallsToBeMade

(IN) Specifies the number of times the **NWDPNtfyMethodListCallback** function is called (the number of methods plus one).

currentCallCount

(IN) Specifies the current call number.

itemReceivedPtr

(IN) Points to the NWDPNtfyDeliveryMethod structure containing the method information.

Return Values

0x00000000	N_SUCCESS

0	
0xFFFFFFFF	N_FAILURE

Remarks

For the *callerDefinedParam* parameter, this function uses the contextual variable to create the following:

a linked list head into which items can be added

a window handle where items can be displayed

any other data or combination of these two steps needed by the callback routine (optional)

For the *totalCallsToBeMade* parameter, the *itemReceivedPtr* parameter is NULL for the last callback.

NCP Calls

None

See Also

NWDPNtfyListMethods

NWDPNtfyModifyProfile

Modifies a notification profile
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyModifyProfile (
   NWDPAccessorRef      accessorRef,
   NWDPNtfyProfileRef   profileRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

profileRef

(IN) Specifies the reference to the modified profile, identifying the alterations to the profile.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

Call either the **NWDPNtfyCreateProfileRefBOP** function or the **NWDPNtfyListProfiles** function to retrieve an existing profile, make modifications to the profile, and then pass the profile as the *profileRef* parameter.

The profile identifier of the *profileRef* parameter must identify an existing registered profile.

If the **NWDPNtfyModifyProfile** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00120004L NWDP_LE_NTFY_NO_SUCH_PROFILE
0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPNtfyCreateProfileRefBOP
NWDPNtfyListProfiles

NWDPNtfyProfileListCallback

Lists the profiles
Local Servers: N/A
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_TYPEDEF_CALLBACK (nint, NWDPNtfyProfileListCallback) (
   NWDPAccessorRef      accessorRef,
   nparam               callerDefinedParam,
   nint                 totalCallsToBeMade,
   nint                 currentCallCount,
   NWDPNtfyProfileRef   profileRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

callerDefinedParam

(IN) Specifies a caller-defined contextual variable utilized by the callback function.

totalCallsToBeMade

(IN) Specifies the number of times the **NWDPNtfyProfileListCallback** function is called (the number of profiles plus one).

currentCallCount

(IN) Specifies the current call number.

profileRef

(IN) Specifies the profile reference which is created for each profile listed.

Return Values

0x00000000	N_SUCCESS

0	
0xFFFFFFFF	N_FAILURE

Remarks

For the *callerDefinedParam* parameter, this function uses the contextual variable to create the following:

a linked list head into which items can be added

a window handle where items can be displayed

any other data or combination of these two steps needed by the callback routine (optional)

For the *totalCallsToBeMade* parameter, the *profileRef* parameter is NULL for the last callback.

If the profile reference needs to exist outside of the callback function, call the **NWDPNtfyUseProfileRef** function to increment the usage count. The user is responsible for calling the **NWDPNtfyReleaseProfileRef** function.

NCP Calls

None

See Also

NWDPNtfyListProfiles

NWDPNtfyProgrammaticCallback

Is called when a notification event is generated
Local Servers: N/A
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_TYPEDEF_CALLBACK (nint, NWDPNtfyProgrammaticCallback) (
   NWDPAccessorRef    accessorRef,
   NWDPPrtRef         printerRef,
   nint32             profileId,
   pNWDPEventReport   eventReportPtr;
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the printer reference associated with the profile.

profileId

(IN) Specifies the identifier of the registered profile that generated the event report.

eventReportPtr

(IN) Points to the NWDPEventReport structure containing the event information.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFF	N_FAILURE

NCP Calls

None

See Also

NWDPNtfyAddProfile

NWDPNtfyPromptListCallback

Lists the prompts
Local Servers: N/A
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_TYPEDEF_CALLBACK (nint, NWDPNtfyPromptListCallback) (
   NWDPAccessorRef           accessorRef,
   nparam                    listCallbackParam2,
   nint                      totalCallsToBeMade,
   nint                      currentCallCount,
   pNWDPNtfyPromptListItem   promptListItemPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

listCallbackParam2

(IN) Specifies a caller-defined contextual variable which is used by the callback function (optional).

totalCallsToBeMade

(IN) Specifies the number of times the **NWDPNtfyPromptListCallback** function is called (the number of prompts plus one).

currentCallCount

(IN) Specifies the current call number.

promptListItemPtr

(IN) Points to the NWDPNtfyPromptListItem structure containing the prompt information.

Return Values

0x00000000	N_SUCCESS

0	
0xFFFFFFFF	N_FAILURE

Remarks

For the *callerDefinedParam* parameter, this function uses the contextual variable to create the following:

a linked list head into which items can be added

a window handle where items can be displayed

any other data or combination of these two steps needed by the callback routine (optional)

For the *totalCallsToBeMade* parameter, the *promptListItemPtr* parameter is NULL for the last callback.

NCP Calls

None

See Also

NWDPNtfyListPrompts

NWDPNtfyReleaseProfileRef

Decrements the usage count of a profile reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyReleaseProfileRef (
   NWDPAccessorRef       accessorRef,
   pNWDPNtfyProfileRef   profileRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

profileRefPtr

(IN) Points to the released profile reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

When the usage count reaches zero, the reference is destroyed and the associated memory is freed.

If the **NWDPNtfyReleaseProfileRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER

NCP Calls

None

See Also

NWDPNtfyCreateProfileRef
NWDPNtfyCreateProfileRefBOP
NWDPNtfyUseProfileRef

NWDPNtfyRemoveDetailEventObject

Removes a detail event object from a profile
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyRemoveDetailEventObject (
   NWDPAccessorRef             accessorRef,
   NWDPNtfyProfileRef          profileRef,
   nuint32                     eventType,
   pNWDPOid                    containingClassOidPtr,
   pNWDPObjectIdentification   containingObjectIdOptionPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

profileRef

(IN) Specifies the reference to the profile where the detail event object is removed.

eventType

(IN) Specifies the event type of the new event object.

containingClassOidPtr

(IN) Points to the containing object class: printer or job.

containingObjectIdOptionPtr

(IN) Points to the object which has events monitored for notification (if NULL is passed, the library places the printer agent name in this field).

Return Values

0x00000000	N_SUCCESS

0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *profileRef* parameter points to an NWDPEventHandlingProfile structure. The **NWDPNtfyRemoveDetailEventObject** function only changes the information contained in this structure. To register this profile with the notification service, call the **NWDPNtfyAddProfile** function or the **NWDPNtfyModifyProfile** function for the existing profiles.

The *eventType* parameter can be the following abstract event types:

NWDP_EVENT_TYPE_ERROR
NWDP_EVENT_TYPE_REPORT
NWDP_EVENT_TYPE_WARNING

The other event type is NWDP_EVENT_TYPE_VALUE_CHANGE.

The OID identified by the *containingClassOidPtr* parameter is the same OID returned in the NWDPEventObjectId structure from the **NWDPNtfyGetSupportedEvents** function.

If the **NWDPNtfyRemoveDetailEventObject** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00120002L NWDP_LE_NTFY_EVENT_OBJ_NOT_FND

NCP Calls

None

See Also

NWDPNtfyAddDetailEventObject
NWDPNtfyAddProfile
NWDPNtfyModifyProfile

NWDPNtfyRemoveEvent

Removes an event from a detail event object
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyRemoveEvent (
   NWDPAccessorRef             accessorRef,
   NWDPNtfyProfileRef          profileRef,
   nuint32                     eventType,
   pNWDPOid                    containingClassOidPtr,
   pNWDPObjectIdentification   containingObjectIdOptionPtr,
   pNWDPOid                     eventOidPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

profileRef

(IN) Specifies the profile containing the detail event object (this event is removed from the event object).

eventType

(IN) Specifies the event type of the new event object.

containingClassOidPtr

(IN) Points to the containing object class: printer or job.

containingObjectIdOptionPtr

(IN) Points to the object which has events monitored for notification (if NULL is passed, the library places the printer agent name in the field).

eventOidPtr

(IN) Points to the event to be removed.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *profileRef* parameter points to an NWDPEventHandlingProfile structure. The **NWDPNtfyRemoveEvent** function only changes the information contained in this structure. To register this profile with the notification service, call the **NWDPNtfyAddProfile** function or the **NWDPNtfyModifyProfile** function for the existing profiles.

The *eventType* parameter can be the following abstract event types:

NWDP_EVENT_TYPE_ERROR
NWDP_EVENT_TYPE_REPORT
NWDP_EVENT_TYPE_WARNING

The other event type is NWDP_EVENT_TYPE_VALUE_CHANGE.

The *eventType*, *containingClassOidPtr*, and *containingObjectIdOptionPtr* parameters determine the existing detail event object from which the event is removed.

If the **NWDPNtfyRemoveEvent** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00120002L NWDP_LE_NTFY_EVENT_OBJ_NOT_FND

NCP Calls

None

See Also

NWDPNtfyAddEvent
NWDPNtfyAddProfile
NWDPNtfyModifyProfile

NWDPNtfyRemoveFilterEventObject

Removes a filter event object from a profile
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyRemoveFilterEventObject (
   NWDPAccessorRef             accessorRef,
   NWDPNtfyProfileRef          profileRef,
   nuint32                     eventType,
   pNWDPOid                    containingClassOidPtr,
   pNWDPObjectIdentification   containingObjectIdOptionPtr,
   pNWDPOid                     filterClassOidPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

profileRef

(IN) Specifies the reference to the profile from which the filter event object is removed.

eventType

(IN) Specifies the event type of the new event object.

containingClassOidPtr

(IN) Points to the containing object class: printer or job.

containingObjectIdOptionPtr

(IN) Points to the object which has events monitored for notification (if NULL is passed, the library places the printer agent name in this field).

filterClassOidPtr

(IN) Points to the removed filter.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *profileRef* parameter points to an NWDPEventHandlingProfile structure. The **NWDPNtfyRemoveFilterEventObject** function changes the information in this structure. To register this profile with the notification service, call the **NWDPNtfyAddProfile** function or the **NWDPNtfyModifyProfile** function for the existing profiles.

The *eventType* parameter can be the following abstract event types:

NWDP_EVENT_TYPE_ERROR
NWDP_EVENT_TYPE_REPORT
NWDP_EVENT_TYPE_WARNING

The other event type is NWDP_EVENT_TYPE_VALUE_CHANGE.

The OID identified by the *containingClassOidPtr* parameter is the same OID returned in the NWDPEventObjectId structure from the **NWDPNtfyGetSupportedEvents** function.

If the **NWDPNtfyRemoveFilterEventObject** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00120002L NWDP_LE_NTFY_EVENT_OBJ_NOT_FND

NCP Calls

None

See Also

NWDPNtfyAddFilterEventObject
NWDPNtfyAddProfile
NWDPNtfyModifyProfile

NWDPNtfyRemoveMethod

Removes a notification method from the notification service
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyRemoveMethod (
   NWDPAccessorRef   accessorRef,
   NWDPNtfyRef       notifyRef,    pNWDPNameOrOid    methodIdPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

notifyRef

(IN) Specifies the notification service with the removed method.

methodIdPtr

(IN) Points to the method identification that is removed.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPNtfyRemoveMethod** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x03060003L NWDP_EC_NOTIFY_RESULT

NCP Calls

None

See Also

NWDPNtfyAddMethod

NWDPNtfyRemoveObjectEventObject

Removes an event object, which is an object type, from a profile
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyRemoveObjectEventObject (
   NWDPAccessorRef             accessorRef,
   NWDPNtfyProfileRef          profileRef,
   nuint32                     eventType,
   pNWDPOid                    containingClassOidPtr,
   pNWDPObjectIdentification   containingObjectIdOptionPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

profileRef

(IN) Specifies the reference to the profile from which the event object of the object type is removed.

eventType

(IN) Specifies the event type of the new event object.

containingClassOidPtr

(IN) Points to the containing object class: printer or job.

containingObjectIdOptionPtr

(IN) Points to the object which has events monitored for notification (if NULL is passed, the library places the printer agent name in this field).

Return Values

0x00000000	N_SUCCESS

0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *profileRef* parameter points to an NWDPEventHandlingProfile structure. The **NWDPNtfyRemoveObjectEventObject** function changes the information in this structure. To register this profile with the notification service, call the **NWDPNtfyAddProfile** function or the **NWDPNtfyModifyProfile** function for the existing profiles.

The *eventType* parameter can be the following abstract event types:

NWDP_EVENT_TYPE_ERROR
NWDP_EVENT_TYPE_REPORT
NWDP_EVENT_TYPE_WARNING

The other event type is NWDP_EVENT_TYPE_VALUE_CHANGE.

The OID identified by the *containingClassOidPtr* parameter is the same OID returned in the NWDPEventObjectId structure from the **NWDPNtfyGetSupportedEvents** function.

If the **NWDPNtfyRemoveObjectEventObject** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00120002L NWDP_LE_NTFY_EVENT_OBJ_NOT_FND

NCP Calls

None

See Also

NWDPNtfyAddObjectEventObject
NWDPNtfyAddProfile
NWDPNtfyModifyProfile

NWDPNtfyRemoveProfile

Removes a profile from the notification service
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyRemoveProfile (
   NWDPAccessorRef     accessorRef,
   NWDPNtfyProfileRef   profileRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

profileRef

(IN) Specifies the reference to the removed profile.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The profile identifier of the *profileRef* parameter specifies which profile is removed.

If the **NWDPNtfyRemoveProfile** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN

NCP Calls

None

See Also

NWDPNtfyAddProfile

NWDPNtfySetDeliveryAddress

Sets the delivery address of a profile
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfySetDeliveryAddress (
   NWDPAccessorRef           accessorRef,
   NWDPNtfyProfileRef        profileRef,
   pNWDPNotifyDeliveryAddr   deliveryAddressPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

profileRef

(IN) Specifies the reference to the profile which has an altered delivery address.

deliveryAddressPtr

(IN) Points to the NWDPNotifyDeliveryAddr structure containing the delivery address(es) for the profile.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID _ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *profileRef* parameter points to the NWDPEventHandlingProfile structure. The **NWDPNtfySetDeliveryAddress** function changes the information contained in this structure.

To register this profile with the notification service, call the **NWDPNtfyAddProfile** function or the **NWDPNtfyModifyProfile** function for the existing profiles.

The **NWDPNtfyListPrompts** function provides the necessary information to complete the notification delivery address(es) for a given method.

NCP Calls

None

See Also

NWDPNtfyAddProfile
NWDPNtfyListPrompts
NWDPNtfyModifyProfile

NWDPNtfySetLanguageId

Sets the language identifier of a profile
Local Servers: nonblocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfySetLanguageId (
   NWDPAccessorRef      accessorRef,
   NWDPNtfyProfileRef   profileRef,
   NWDPLanguageId       languageId);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

profileRef

(IN) Specifies the profile which has an altered language identifier.

languageId

(IN) Specifies the new language used for notifications (pass NWDP_LID_DEFAULT to use the language identified by the environment variable NWLANGUAGE).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *profileRef* parameter points to the NWDPEventHandlingProfile structure. The **NWDPNtfySetLanguageId** function only changes the information contained in this structure. To register this profile with the notification service, call the **NWDPNtfyAddProfile** function or the **NWDPNtfyModifyProfile** function for the existing profiles.

NCP Calls

None

See Also

NWDPNtfyAddProfile
NWDPNtfyModifyProfile

NWDPNtfySetMethodId

Sets the method identifier of a profile
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfySetMethodId (
   NWDPAccessorRef      accessorRef,
   NWDPNtfyProfileRef   profileRef,
   pNWDPNameOrOid       methodIdPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

profileRef

(IN) Specifies the reference to the profile which has an altered method identifier.

methodIdPtr

(IN) Points to the new method identifier.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *profileRef* parameter points to the NWDPEventHandlingProfile

The *profileRef* parameter points to the NWDPEventHandlingProfile structure. The **NWDPNtfySetMethodId** function only changes the information contained in this structure. To register this profile with the notification service, call the **NWDPNtfyAddProfile** function or the **NWDPNtfyModifyProfile** function for the existing profiles.

NCP Calls

None

See Also

NWDPNtfyAddProfile
NWDPNtfyModifyProfile

NWDPNtfySetSupplierId

Sets the supplier identifier of a profile
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfySetSupplierId (
   NWDPAccessorRef      accessorRef,
   NWDPNtfyProfileRef   profileRef,
   pNWDPOctetString     supplierIdPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

profileRef

(IN) Specifies the reference to the profile which has an altered supplier identifier.

supplierIdPtr

(IN) Points to the new supplier identifier.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *profileRef* parameter points to the NWDPEventHandlingProfile

The *profileRef* parameter points to the NWDPEventHandlingProfile structure. The **NWDPNtfySetSupplierId** function only changes the information contained in this structure.

To register this profile with the notification service, call the **NWDPNtfyAddProfile** function or the **NWDPNtfyModifyProfile** function for the existing profiles.

For the *supplierIdPtr* parameter, the new supplier identifier is a caller-defined parameter. The entered information is stored in the profile and is returned whenever the profiles are retrieved.

NCP Calls

None

See Also

NWDPNtfyAddProfile
NWDPNtfyModifyProfile

NWDPNtfyUseProfileRef

Increments the usage count of a profile reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyUseProfileRef (
   NWDPAccessorRef      accessorRef,
   NWDPNtfyProfileRef   profileRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

profileRef

(IN) Specifies the reference to the profile which has an incremented usage count.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPNtfyUseProfileRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER

NCP Calls

None

See Also

NWDPNtfyCreateProfileRef
NWDPNtfyReleaseProfileRef

NWDPNtfyValidateMethodInfoRef

Validates a method information reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyValidateMethodInfoRef (
   NWDPAccessorRef          accessorRef,
   NWDPNtfyMethodInfoRef    methodInfoRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

methodInfoRef

(IN) Specifies the validated method information reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The **NWDPNtfyValidateMethodInfoRef** function verifies that the method information reference is associated with the *accessorRef* parameter when the application has received it from an untrusted third party.

If the **NWDPNtfyValidateMethodInfoRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER

NCP Calls

None

NWDPNtfyValidateProfileRef

Validates a profile reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyValidateProfileRef (
   NWDPAccessorRef      accessorRef,
   NWDPNtfyProfileRef   profileRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

profileRef

(IN) Specifies the validated profile reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The **NWDPNtfyValidateProfileRef** function verifies that the profile reference is associated with the *accessorRef* parameter when the application has received it from an untrusted party.

If the **NWDPNtfyValidateProfileRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER

NCP Calls

None

NWDPNtfyValidateRef

Validates a notification reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_not.h>

N_EXTERN_LIBRARY (nint) NWDPNtfyValidateRef (
   NWDPAccessorRef   accessorRef,
   NWDPNtfyRef       notifyRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

profileRef

(IN) Specifies the notification reference to validate.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The **NWDPNtfyValidateProfileRef** function verifies that the notification reference is associated with the *accessorRef* parameter when the application has received it from an untrusted third party.

If the **NWDPNtfyValidateProfileRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER

NCP Calls

None

NWDPOidCmp

Compares two OIDs for equality
Local Servers: nonblocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_oid.h>

N_EXTERN_LIBRARY (nint) NWDPOidCmp (
   NWDPAccessorRef   accessorRef,
   pNWDPOid          oid1Ptr,
   pNWDPOid          oid2Ptr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

oid1Ptr

(IN) Points to the first OID.

oid2Ptr

(IN) Points to the second OID.

Return Values

0x00000000	N_SUCCESS
0x00000001	NWDP_RC_WARNING
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPOIDCMP** function returns N_SUCCESS, the OIDs are equal.

equal.

If the **NWDPOIDCMP** function returns NWDP_RC_WARNING, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00140004L NWDP_LE_COMPARE_RESULT_NE (indicating the OIDs are unequal)

If the **NWDPOIDCMP** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x10000001 NWDP_EC_INVALID_PARAMETER

0x000000EL NWDP_LE_OID_INVALID_ASN1_TYPE

NCP Calls

None

NWDPOidCreateRefBasedOnASCII

Returns an object identifier reference based on the decimal ASCII string with periods used for separators

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_oid.h>

N_EXTERN_LIBRARY (nint) NWDPOidCreateRefBasedOnASCII (
   NWDPAccessorRef   accessorRef,
   pnstr             oidASCIIStringPtr,
   pNWDPOidRef       oidRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

oidASCIIStringPtr

(IN) Points to the string to be converted to binary and stored within the reference structure.

oidRefPtr

(OUT) Points to the NWDPOidRef containing the resulting object identifier reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFF	N_FAILURE
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR

Remarks

Remarks

If the **NWDPOidCreateRefBasedOnASCII** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER
0x030A0001L NWDP_EC_NO_MEMORY
0x000D0001L NWDP_LE_OID_INVALID_ASCII

NCP Calls

None

NWDPOidCreateRefBasedOnOid

Returns an object identifier reference based on binary data being already encoded

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_oid.h>

N_EXTERN_LIBRARY (nint) NWDPOidCreateRefBasedOnOid (
   NWDPAccessorRef   accessorRef,
   pNWDPOid          oidPtr,
   pNWDPOidRef       oidRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

oidPtr

(IN) Points to NWDPOid structure containing a string of encoded bytes to be made into a reference.

oidRefPtr

(OUT) Points to the NWDPOidRef.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFF	N_FAILURE
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR

Remarks

The **NWDPOidCreateRefBasedOnOid** function allows a subsystem to receive an object identifier in a packet and create a library-recognizable object identifier reference. The resulting reference does not depend on the further existence of the data pointed to by the *oidPtr* parameter.

The first byte of the *oidPtr* parameter must contain a 6, and the second byte must contain the number of bytes in the remainder of the object identifier string according to ASN1 BER.

If the **NWDPOidCreateRefBasedOnOid** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER
0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

NWDPOidCreateRefBasedOnString

Returns an object identifier reference based on the natural language string, the context given, and the type of object preferred

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_oid.h>

N_EXTERN_LIBRARY (nint) NWDPOidCreateRefBasedOnString (
   NWDPAccessorRef   accessorRef,
   pnstr             oidInterpretPtr,
   NWDPLanguageId    languageId,
   NWDPOidKind       oidKind,
   pNWDPOidRef       oidRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

oidInterpretPtr

(IN) Points to the string to be interpreted.

languageId

(IN) Specifies the NWDPLanguageId enumeration.

oidKind

(IN) Specifies the NWDPOidKind enumeration.

oidRefPtr

(OUT) Points to the NWDPOidRef containing the resulting object identifier reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFF	N_FAILURE

FF	
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR

Remarks

The *languageId* parameter can have the following values:

-1 NWDP_LID_DEFAULT
0 NWDP_LID_CANADIAN_FRENCH
1 NWDP_LID_CHINESE (simplified)
2 NWDP_LID_DANISH
3 NWDP_LID_DUTCH
4 NWDP_LID_ENGLISH
5 NWDP_LID_FINNISH
6 NWDP_LID_ FRENCH
7 NWDP_LID_GERMAN
8 NWDP_LID_ITALIAN
9 NWDP_LID_JAPANESE
10 NWDP_LID_KOREAN
11 NWDP_LID_NORWEGIAN
12 NWDP_LID_PORTUGUESE (Brazil)
13 NWDP_LID_RUSSIAN
14 NWDP_LID_SPANISH (Latin America)
15 NWDP_LID_SWEDISH
16 NWDP_LID_CHINESE_TRAD (traditional)
17 NWDP_LID_POLISH
18 NWDP_LID_PORTUGUESE_PORT (Portugal)
19 NWDP_LID_SPANISH_SPAIN (Spain)
20 NWDP_LID_HUNGARIAN
21 NWDP_LID_CZECH
98 NWDP_LID_FJP (Fake Japanese)
99 NWDP_LID_FEU (Fake European)

New *languageId* parameter values are added as new contexts are defined. Contexts are not limited to natural languages; however, only natural languages can be used for the **NWDPOidCreateRefBasedOnString** function. If the NWDP_ITPC_XXXX constant is a negative value, its use is invalid and the error NWDP_LE_OID_NOT_LANG_CONTEXT is returned by the NWDPLibErrorMac macro.

The *oidKind* parameter can have the following values:

0 NWDP_OK_OBJECT_CLASS
1 NWDP_OK_VALUE_CLASS
2 NWDP_OK_ATTRIBUTE
3 NWDP_OK_VALUE

Since mappings of textual interpretations to object IDs are one-to-many, the *oidKind* parameter is necessary, for example, to determine whether the original DPA Object identifier name had a prefix of id-oc-, id-vc-, id-att-, or id-val-.

If the **NWDPOidCreateRefBasedOnString** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER
0x01000002L NWDP_EC_PARAM_VAL_UNRECOGNIZED
0x01000004L NWDP_EC_STDIO
0x01000005L NWDP_EC_NDS
0x030A0001L NWDP_EC_NO_MEMORY
0x000D0005L NWDP_LE_OID_STRING_NOT_FOUND
0x000D0006L NWDP_LE_OID_NO_STRING_THAT_KIND
0x000D000BL NWDP_LE_OID_NOT_LANG_CONTEXT

NCP Calls

None

NWDPOidGetAttrCharacteristics

Gets characteristics of the attribute whose OID is input
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_oid.h>

N_EXTERN_LIBRARY (nint) NWDPOidGetAttrCharacteristics (
   NWDPAccessorRef   accessorRef,
   NWDPOidRef        oidRef,
   pNWDPAVTEnum      attrTypeEnumPtr,
   pnbool            isMultiValuedPtr,
   pnbool            hasEnumeratedValuesPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

oidRef

(IN) Points to the OID of an attribute.

attrTypeEnumPtr

(OUT) Points to an NWDPAVTEnum value representing the attribute's syntax.

isMultiValuedPtr

(OUT) Points to a boolean (if equal to N_TRUE, it implies that the attribute is multivalued).

hasEnumeratedValuesPtr

(OUT) Points to a boolean (if equal to N_TRUE, it implies that the syntax enum is NWDP_AVT_ENUMERATION and the value of the attribute can be interpreted using the **NWDPOidInterpretRefValue** function).

Return Values

0x00000	N_SUCCESS

0x00000000	N_SUCCESS

Remarks

One of the purposes of the **NWDPOidGetAttrCharacteristics** function is to aid in the interpretation of the attributes and their values as human readable text without hard-coding an endless set of special cases.

NCP Calls

None

See Also

NWDPOidInterpretRefValue

NWDPOidInterpretRef

Interprets the specified *oidRef* parameter based on the context given and returns the interpretation in the buffer provided by the user

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_oid.h>

N_EXTERN_LIBRARY (nint) NWDPOidInterpretRef (
   NWDPAccessorRef         accessorRef,
   NWDPOidRef              oidRef,
   NWDPLanguageId          languageId,
   nuint                   sizeOfBuffer,
   pnuint8                 sizeOfResultPtr,
   pNWDPInterpretFormat    formatIdPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

oidRef

(IN) Specifies the NWDPOidRef containing the object identifier reference.

languageId

(IN) Specifies the NWDPLanguageId enumeration value used to produce text in the implied language.

sizeOfBuffer

(IN) Specifies the number of bytes provided for receiving the interpretation.

oidInterpretBufferPtr

(OUT) Points to the receiving buffer having the minimum byte size specified by the *sizeOfBuffer* parameter.

sizeOfResultPtr

(OUT) Points to the number of bytes used to store the interpretation.

formatIdPtr

(OUT) Points to the NWDPInterpretFormat enumeration describing the interpretation format:

0 NWDP_ITPF_STRING
1 NWDP_ITPF_NUINT32_AND_STRING

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFF	N_FAILURE
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR

Remarks

The buffer syntax is also returned by the **NWDPOidInterpretRef** function so the user can deal with different interpretation syntaxes without trying to interpret the buffer contents directly.

The *languageId* parameter can have the following values:

-1 NWDP_LID_DEFAULT
0 NWDP_LID_CANADIAN_FRENCH
1 NWDP_LID_CHINESE (simplified)
2 NWDP_LID_DANISH
3 NWDP_LID_DUTCH
4 NWDP_LID_ENGLISH
5 NWDP_LID_FINNISH
6 NWDP_LID_ FRENCH
7 NWDP_LID_GERMAN
8 NWDP_LID_ITALIAN
9 NWDP_LID_JAPANESE
10 NWDP_LID_KOREAN
11 NWDP_LID_NORWEGIAN
12 NWDP_LID_PORTUGUESE (Brazil)
13 NWDP_LID_RUSSIAN
14 NWDP_LID_SPANISH (Latin America)
15 NWDP_LID_SWEDISH
16 NWDP_LID_CHINESE_TRAD (traditional)
17 NWDP_LID_POLISH
18 NWDP_LID_PORTUGUESE_PORT (Portugal)
19 NWDP_LID_SPANISH_SPAIN (Spain)
20 NWDP_LID_HUNGARIAN

21 NWDP_LID_CZECH
98 NWDP_LID_FJP (Fake Japanese)
99 NWDP_LID_FEU (Fake European)

New *languageId* parameter values are added as new contexts are defined. Contexts are not limited to natural languages.

New *formatIdPtr* parameter values is added as new buffer formats are defined. Buffer formats are not limited to strings or groups of strings.

If the **NWDPOidInterpretRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER
0x01000002L NWDP_EC_PARAM_VAL_UNRECOGNIZED
0x01000004L NWDP_EC_STDIO
0x030A0001L NWDP_EC_NO_MEMORY
0x000A0002L NWDP_LE_MEMORY_NOT_MALLOCED
0x000A0006L NWDP_LE_RESULT_BUFFER_TOO_SMALL
0x000D0003L NWDP_LE_OID_NOT_IN_DATABASE
0x000D0004L NWDP_LE_OID_UNSUPPORTED_FORMAT
0x000D0007L NWDP_LE_OID_MSG_FILE_NOT_FOUND

NCP Calls

None

NWDPOidInterpretRefAsASCII

Interprets the specified *oidRef* parameter as an ASCII decimal string with periods between fields, a terminating NULL (`\0'), and no other delimiters or text

Local Servers: nonblocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_oid.h>

N_EXTERN_LIBRARY (nint) NWDPOidInterpretRefAsASCII (
   NWDPAccessorRef   accessorRef,
   NWDPOidRef        oidRef,
   nuint             sizeOfBuffer,
   pnstr             oidASCIIBufferPtr,
   pnuint            sizeOfResultPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

oidRef

(IN) Specifies the NWDPOidRef containing the object identifier reference.

sizeOfBuffer

(IN) Specifies the number of bytes provided to receive the interpretation.

oidASCIIBufferPtr

(OUT) Points to the receiving buffer having the minimum byte size specified by the *sizeOfBuffer* parameter.

sizeOfResultPtr

(OUT) Points to the number of bytes used to store the interpretation.

Return Values

0x0000000	N_SUCCESS

0x00000000	N_SUCCESS
0xFFFFFFFF	N_FAILURE
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR

Remarks

Calling the **NWDPOidInterpretRefAsASCII** function guarantees that all object IDs represented by the *oidRef* parameter are interpreted as text regardless of the language context or the newness of the object identifier and its interpretation.

If the **NWDPOidInterpretRefAsASCII** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER
0x000A0006L NWDP_LE_RESULT_BUFFER_TOO_SMALL

NCP Calls

None

NWDPOidInterpretRefValue

Interprets the specified *oidRef* parameter for an attribute and its *enumValue* parameter based on the context given and returns the interpretation in the buffer provided by the user

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_oid.h>

N_EXTERN_LIBRARY (nint) NWDPOidInterpretRefValue (
   NWDPAccessorRef        accessorRef,
   NWDPOidRef             oidRef,
   nint32                 enumValue,
   NWDPLanguageId         languageId,
   nuint                  sizeOfBuffer,
   pnuint8                oidInterpretBufferPtr
   pnuint                 sizeOfResultPtr,
   pNWDPInterpretFormat   formatIdPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

oidRef

(IN) Specifies the NWDPOidRef containing the object identifier reference of an attribute.

enumValue

(IN) Specifies the integer value of the attribute to be interpreted.

languageId

(IN) Specifies the NWDPLanguageId enumeration to identify the target language of the interpretation.

sizeOfBuffer

(IN) Specifies the number of bytes provided for receiving the interpretation.

oidInterpretBufferPtr

(OUT) Points to the receiving buffer having the minimum byte size

specified by the *sizeOfBuffer* parameter.

sizeOfResultPtr

(OUT) Points to the number of bytes used to store the interpretation (optional).

formatIdPtr

(OUT) Points to the NWDPInterpretFormat enumeration describing the interpretation format:

0 NWDP_ITPF_STRING
1 NWDP_ITPF_NUINT32_AND_STRING

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFF	N_FAILURE
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR

Remarks

The buffer syntax is also returned by the **NWDPOidInterpretRefValue** function so the user can deal with different interpretation syntaxes without trying to interpret the buffer contents directly.

The *languageId* parameter can have the following values:

-1 NWDP_LID_DEFAULT
0 NWDP_LID_CANADIAN_FRENCH
1 NWDP_LID_CHINESE (simplified)
2 NWDP_LID_DANISH
3 NWDP_LID_DUTCH
4 NWDP_LID_ENGLISH
5 NWDP_LID_FINNISH
6 NWDP_LID_ FRENCH
7 NWDP_LID_GERMAN
8 NWDP_LID_ITALIAN
9 NWDP_LID_JAPANESE
10 NWDP_LID_KOREAN
11 NWDP_LID_NORWEGIAN
12 NWDP_LID_PORTUGUESE (Brazil)
13 NWDP_LID_RUSSIAN
14 NWDP_LID_SPANISH (Latin America)

15 NWDP_LID_SWEDISH
16 NWDP_LID_CHINESE_TRAD (traditional)
17 NWDP_LID_POLISH
18 NWDP_LID_PORTUGUESE_PORT (Portugal)
19 NWDP_LID_SPANISH_SPAIN (Spain)
20 NWDP_LID_HUNGARIAN
21 NWDP_LID_CZECH
98 NWDP_LID_FJP (Fake Japanese)
99 NWDP_LID_FEU (Fake European)

New *languageId* parameter values are added as new contexts are defined. Contexts are not limited to natural language.

New *formatIdPtr* parameter values are added as new buffer formats are defined. Buffer formats are not limited to strings or groups of strings.

If the **NWDPOidInterpretRefValue** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER
0x01000002L NWDP_EC_PARAM_VAL_UNRECOGNIZED
0x01000004L NWDP_EC_STDIO
0x030A0001L NWDP_EC_NO_MEMORY
0x000A0002L NWDP_LE_MEMORY_NOT_MALLOCED
0x000A0006L NWDP_LE_RESULT_BUFFER_TOO_SMALL
0x000D0003L NWDP_LE_OID_NOT_IN_DATABASE
0x000D0004L NWDP_LE_OID_UNSUPPORTED_FORMAT
0x000D0007L NWDP_LE_OID_MSG_FILE_NOT_FOUND

NCP Calls

None

NWDPOidInterpretRefWithClosest

Interprets the specified *oidRef* parameter based on the context given and returns the interpretation in the buffer provided by the user

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_oid.h>

N_EXTERN_LIBRARY (nint) NWDPOidInterpretRefWithClosest (
   NWDPAccessorRef        accessorRef,
   NWDPOidRef             oidRef,
   NWDPLanguageId         languageId,
   nuint                  sizeOfBuffer,
   pnuint8                oidInterpretBufferPtr,
   pnuint                 sizeOfResultPtr,
   pNWDPInterpretFormat   formatIdPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

oidRef

(IN) Specifies the NWDPOidRef containing the object identifier reference.

languageId

(IN) Specifies the NWDPLanguageId enumeration.

sizeOfBuffer

(IN) Specifies the number of bytes provided for receiving the interpretation.

oidInterpretBufferPtr

(OUT) Points to the receiving buffer having the minimum byte size specified by the *sizeOfBuffer* parameter.

sizeOfResultPtr

(OUT) Points to the number of bytes used to store the interpretation.

formatIdPtr

(OUT) Points to the NWDPInterpretFormat enumeration describing the interpretation format:

0 NWDP_ITPF_STRING
1 NWDP_ITPF_NUINT32_AND_STRING

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFF	N_FAILURE
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR

Remarks

The **NWDPOidInterpretRefWithClosest** function extends what is accomplished by the **NWDPOidInterpretRef** function. The **NWDPOidInterpretRefWithClosest** function calls the **NWDPOidInterpretRef** function. If there is no exact match, the object identifier value is shrunk to the previous level of the numbering scheme to see if the object identifier is recognized. This process continues until a match occurs, or until the object identifier has no recognizable subcomponent.

In the event there is no recognizable subcomponent, the **NWDPOidInterpretRefAsASCII** function is called and the return value contained in the *formatIdPtr* parameter is NWDP_ITPF_STRING.

The buffer syntax is also returned by the **NWDPOidInterpretRefWithClosest** function so the user can deal with different interpretation syntaxes without trying to interpret the buffer contents directly.

The *languageId* parameter can have the following values:

-1 NWDP_LID_DEFAULT
0 NWDP_LID_CANADIAN_FRENCH
1 NWDP_LID_CHINESE (simplified)
2 NWDP_LID_DANISH
3 NWDP_LID_DUTCH
4 NWDP_LID_ENGLISH
5 NWDP_LID_FINNISH
6 NWDP_LID_ FRENCH
7 NWDP_LID_GERMAN
8 NWDP_LID_ITALIAN

9 NWDP_LID_JAPANESE
10 NWDP_LID_KOREAN
11 NWDP_LID_NORWEGIAN
12 NWDP_LID_PORTUGUESE (Brazil)
13 NWDP_LID_RUSSIAN
14 NWDP_LID_SPANISH (Latin America)
15 NWDP_LID_SWEDISH
16 NWDP_LID_CHINESE_TRAD (traditional)
17 NWDP_LID_POLISH
18 NWDP_LID_PORTUGUESE_PORT (Portugal)
19 NWDP_LID_SPANISH_SPAIN (Spain)
20 NWDP_LID_HUNGARIAN
21 NWDP_LID_CZECH
98 NWDP_LID_FJP (Fake Japanese)
99 NWDP_LID_FEU (Fake European)

New *languageId* parameter values are added as new contexts are defined. Contexts are not limited to natural languages.

New *formatIdPtr* parameter values are added as new buffer formats are defined. Buffer formats are not limited to strings or groups of strings.

If the **NWDPOidInterpretRefWithClosest** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER
0x01000002L NWDP_EC_PARAM_VAL_UNRECOGNIZED
0x01000004L NWDP_EC_STDIO
0x030A0001L NWDP_EC_NO_MEMORY
0x000A0002L NWDP_LE_MEMORY_NOT_MALLOCED
0x000A0006L NWDP_LE_RESULT_BUFFER_TOO_SMALL
0x000D0004L NWDP_LE_OID_UNSUPPORTED_FORMAT
0x000D0007L NWDP_LE_OID_MSG_FILE_NOT_FOUND

NCP Calls

None

NWDPOidListRefValues

Lists the interpretation values possible with an enumerated object identifier based on the context given and returns all of the possible interpretations using the standard callback mechanism provided by the user

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_oid.h>

N_EXTERN_LIBRARY (nint) NWDPOidListRefValues (
   NWDPAccessorRef            accessorRef,
   NWDPOidRef                 oidRef,
   NWDPLanguageId             languageId,
   NWDPOidValueListCallback   listCallback,
   nparam                      callerDefinedParam);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

oidRef

(IN) Specifies the NWDPOidRef containing the object identifier reference for the attribute whose syntax is NWDP_AVT_ENUMERATION.

languageId

(IN) Specifies the NWDPLanguageId containing an enumeration.

listCallBack

(IN) Specifies the **NWDPOidValueListCallback** function to be called with each value.

callerDefinedParam

(IN) Specifies a user-defined parameter passed to the **NWDPOidValueListCallback** function containing application-specific data.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFF	N_FAILURE
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR

Remarks

The *languageId* parameter can have the following values:

-1 NWDP_LID_DEFAULT
0 NWDP_LID_CANADIAN_FRENCH
1 NWDP_LID_CHINESE (simplified)
2 NWDP_LID_DANISH
3 NWDP_LID_DUTCH
4 NWDP_LID_ENGLISH
5 NWDP_LID_FINNISH
6 NWDP_LID_ FRENCH
7 NWDP_LID_GERMAN
8 NWDP_LID_ITALIAN
9 NWDP_LID_JAPANESE
10 NWDP_LID_KOREAN
11 NWDP_LID_NORWEGIAN
12 NWDP_LID_PORTUGUESE (Brazil)
13 NWDP_LID_RUSSIAN
14 NWDP_LID_SPANISH (Latin America)
15 NWDP_LID_SWEDISH
16 NWDP_LID_CHINESE_TRAD (traditional)
17 NWDP_LID_POLISH
18 NWDP_LID_PORTUGUESE_PORT (Portugal)
19 NWDP_LID_SPANISH_SPAIN (Spain)
20 NWDP_LID_HUNGARIAN
21 NWDP_LID_CZECH
98 NWDP_LID_FJP (Fake Japanese)
99 NWDP_LID_FEU (Fake European)

New *languageId* parameter values are added as new languages are defined. Languages are not limited to natural languages.

If the **NWDPOidListRefValues** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER
0x01000002L NWDP_EC_PARAM_VAL_UNRECOGNIZED

0x01000004L NWDP_EC_STDIO
0x030A0001L NWDP_EC_NO_MEMORY
0x000A0002L NWDP_LE_MEMORY_NOT_MALLOCED
0x000A0006L NWDP_LE_RESULT_BUFFER_TOO_SMALL
0x000D0004L NWDP_LE_OID_UNSUPPORTED_FORMAT
0x000D0007L NWDP_LE_OID_MSG_FILE_NOT_FOUND

NCP Calls

None

NWDPOidMakeOidPtrFromOidRef

Makes an OID pointer from an OID reference
Local Servers: nonblocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_oid.h>

N_EXTERN_LIBRARY (pNWDPOid) NWDPOidMakeOidPtrFromOidRef (
   NWDPOidRef   oidRef);
```

Parameters

oidRef

(IN) Points to the OID structure containing the NWDPOid.

Return Values

The **NWDPOidMakeOidPtrFromOidRef** function returns a pointer to the OID hidden within the reference.

Remarks

The **NWDPOidMakeOidPtrFromOidRef** function cannot fail.

NCP Calls

None

NWDPOidReleaseRef

Releases the reference to an Object ID
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_oid.h>

N_EXTERN_LIBRARY (nint) NWDPOidReleaseRef (
   NWDPAccessorRef   accessorRef,
   pNWDPOidRef       oidRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

oidRefPtr

(IN) Points to the NWDPOidRef containing the reference to be released.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFF	N_FAILURE
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR

Remarks

If the usage count reaches zero, the *oidRefPtr* parameter is freed and the OID reference pointed to is set to zero.

If the **NWDPOidReleaseRef** function returns N_FAILURE, the NWDPLibErrorMac (*accessorRef)* contains the following:

NWDPLibErrorMac (*accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER

NCP Calls

None

NWDPOidSetListCallback

Lists the Object Identifiers (OIDs) in a set
Local Servers: N/A
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_oid.h>

N_TYPEDEF_CALLBACK (nint, NWDPOidSetListCallback) (
   NWDPAccessorRef   accessorRef,
   nparam            callerDefinedParam,
   nint              totalCallsToBeMade,
   nint              currentCallCount,
   pNWDPOid          oidPtr);
```

Parameters

accessorRef

Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

callerDefinedParam

(IN) Specifies a caller-defined contextual variable.

totalCallsToBeMade

(IN) Specifies the number of times the **NWDPOSListOids** function calls the **NWDPOidSetListCallback** function (the number of OIDs plus one).

currentCallCount

(IN) Specifies the current call number.

oidPtr

(IN) Points to the currently listed OID.

Return Values

0x00000000	N_SUCCESS
0xFFFFFF	N_FAILURE

FF	

Remarks

For the *callerDefinedParam* parameter, this function uses the contextual variable to create the following:

a linked list head into which items can be added

a window handle where items can be displayed

any other data or combination of these two steps needed by the callback routine (optional)

For the *totalCallsToBeMade* parameter, the *oidPtr* parameter is set to NULL for the last callback.

NCP Calls

None

See Also

NWDPOSListOids

NWDPOidValidateRef

Validates the origin of the OID reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_oid.h>

N_EXTERN_LIBRARY (nint) NWDPOidValidateRef (
   NWDPAccessorRef   accessorRef,
   NWDPOidRef        oidRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

oidRef

(IN) Points to the OID.

Return Values

0x00000000	N_SUCCESS

Remarks

The **NWDPOidValidateRef** function verifies that the OID reference is associated with the *accessorRef* parameter when the application has received it from an untrusted party.

NCP Calls

None

NWDPOidValueListCallback

Lists the possible values for an enumeration attribute
Local Servers: N/A
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_oid.h>

N_TYPEDEF_CALLBACK (nint, NWDPOidValueListCallback) (
   NWDPAccessorRef          accessorRef,
   nparam                   callerDefinedParam,
   nint                     totalCallsToBeMade,
   nint                     currentCallCount,
   pNWDPOidValueListItem    itemReceivedPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

callerDefinedParam

(IN) Specifies a caller-defined contextual variable.

totalCallsToBeMade

(IN) Specifies the number of times the **NWDPOidListRefValues** function calls the **NWDPOidValueListCallback** function (the number of OIDs plus one).

currentCallCount

(IN) Specifies the current call number.

itemReceivedPtr

(IN) Points to the structure containing the enumeration value (integer) and to the interpretation of the value in the specified language.

Return Values

0x00000000	N_SUCCESS

NCP Calls

None

See Also

NWDPOidListRefValues

NWDPOSAddOid

Adds an Object Identifier (OID) to an OID set
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_oid.h>

N_EXTERN_LIBRARY (nint) NWDPOSAddOid (
   NWDPAccessorRef   accessorRef,
   NWDPOidSetRef     oidSetRef,
   pNWDPOid          oidPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

oidSetRef

(IN) Specifies the reference to the OID set.

oidPtr

(IN) Points to the NWDPOid structure containing the OID to be added to the set.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPOSAddOid** function returns N_FAILURE, the

If the **NWDPOSAddOid** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPOSRemoveOid

NWDPOSCreateRef

Creates an OID set reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_oid.h>

N_EXTERN_LIBRARY (nint) NWDPOSCreateRef (
   NWDPAccessorRef   accessorRef,
   nuint             sizeOfOidSet,
   pNWDPOidSetRef    oidSetRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

sizeOfOidSet

(IN) Specifies the number of OIDs needing initially allocated memory.

oidSetRefPtr

(OUT) Points to the NWDPOidSetRef, the resulting OID set reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the OID set expands beyond the number specified by the *sizeOfOidSet* parameter, the library allocates more memory. If you are unsure of the

parameter, the library allocates more memory. If you are unsure of the OID set size, pass NWDP_OID_SET_SIZE.

If the **NWDPOSCreateRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPOSReleaseRef
NWDPOSUseRef

NWDPOSListOids

Lists the OIDs of an OID set
Local Servers: nonblocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_oid.h>

N_EXTERN_LIBRARY (nint) NWDPOSListOids (
   NWDPAccessorRef          accessorRef,
   NWDPOidSetRef            oidSetRef,
   NWDPOidSetListCallback   listCallback,
   nparam                   callerDefinedParam);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

oidSetRef

(IN) Specifies the reference to the set of OIDs that are listed.

listCallback

(IN) Specifies the application-supplied callback function that the **NWDPOSListOids** function calls for each OID.

callerDefinedParam

(IN) Specifies a caller-defined parameter, containing application-specific data, that is passed to the callback function (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

FF	

Remarks

The **NWDPOSListOids** function is generally nonblocking, except when the associated callback function is blocking, then the **NWDPOSListOids** function is also blocking.

If the **NWDPOSListOids** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x000A0005L NWDP_LE_UNKNOWN_CALLBACK_ERR

NCP Calls

None

NWDPOSReleaseRef

Decrements the usage count for an OID set reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_oid.h>

N_EXTERN_LIBRARY (nint) NWDPOSReleaseRef (
   NWDPAccessorRef   accessorRef,
   pNWDPOidSetRef    oidSetRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

OidSetRefPtr

(IN) Points to the OID set reference that is released.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

When the usage count reaches zero, the reference is destroyed and the associated memory is freed.

If the **NWDPOSReleaseRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER

NCP Calls

None

See Also

NWDPOSCreateRef
NWDPOSUseRef

NWDPOSRemoveOid

Removes an OID from an OID set
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_oid.h>

N_EXTERN_LIBRARY (nint) NWDPOSRemoveOid (
   NWDPAccessorRef   accessorRef,
   NWDPOidSetRef     oidSetRef,
   pNWDPOid          oidPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

oidSetRef

(IN) Specifies the reference to the OID set.

oidPtr

(IN) Points to the NWDPOid structure containing the OID removed from the set.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

It is not necessary to remove an OID from a set before you release the set,

It is not necessary to remove an OID from a set before you release the set, unless another user of the set expects it to be removed. All memory allocated for the OID set and its elements will be freed with the final release of the reference.

If the **NWDPOSRemoveOid** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x000D000DL NWDP_LE_OID_NOT_IN_SET

NCP Calls

None

See Also

NWDPOSAddOid

NWDPOSUseRef

Increments the usage count for an OID set reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_oid.h>

N_EXTERN_LIBRARY (nint) NWDPOSUseRef (
   NWDPAccessorRef   accessorRef,
   NWDPOidSetRef     oidSetRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

oidSetRef

(IN) Specifies the OID set reference that has its usage count incremented.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPOSUseRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER

NCP Calls

None

See Also

NWDPOSCreateRef
NWDPOSReleaseRef

NWDPOSValidateRef

Validates an OID set reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_oid.h>

N_EXTERN_LIBRARY (nint) NWDPOSValidateRef (
   NWDPAccessorRef   accessorRef,
   NWDPOidSetRef     oidSetRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

oidSetRef

(IN) Specifies the OID set reference to validate.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The **NWDPOSValidateRef** function verifies that the OID set reference is associated with the *accessorRef* parameter when the application has received it from an untrusted party.

If the **NWDPOSValidateRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER

NCP Calls

None

NWDPPrtAddInstalledPrinter

Adds a controlled-access printer to the user's installed list (shortlist)
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtAddInstalledPrinter (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        printerRef,
   pnstr16           jobconfigName16Ptr,
   pnstr16           label16Ptr,
   pNWDPPrtRef       localPrinterRefPtr,
   nuint32           flags);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the reference to the controlled-access printer that is installed on the user's installed list (this printer reference must be created based on its FQN).

jobconfigName16Ptr

(IN) Points to the name of a configuration (a Unicode string) on the NDPS printer object identified by the *printerRef* parameter and used to install the printer (optional).

label16Ptr

(IN) Points to the label assigned to the installed printer (this is a Unicode string).

localPrinterRefPtr

(OUT) Points to the NWDPPrtRef, the resulting installed printer reference (optional).

flags

(IN) Specify the procedure for the library to follow for printer references.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If a NDPS printer object configuration is preferred, the *jobConfigName16Ptr* parameter points to one of these configurations. If the default configuration of the printer is preferred, pass NULL for the *jobConfigName16Ptr* parameter. The *printerRef* parameter identifies the NDPS printer object in NDS.

Passing NULL for the *jobConfigName16Ptr* parameter is identical to calling the **NWDPPrtAddInstalledPrtWConfig** function with the *jobConfigRef* parameter and the *docConfigRef* parameter both NULL.

For the *flags* parameter, pass one of the following:

NWDP_PRT_FLG_NO_REF
NWDP_PRT_FLG_CONVERT_REF
NULL

For the *flags* parameter, the printer references are affected as follows:

NWDP_PRT_FLG_NO_REF (The *localPrinterRefPtr* parameter should be NULL.)
NWDP_PRT_FLG_CONVERT_REF (The *localPrinterRefPtr* parameter should be NULL. The *printerRef* parameter converts to an installed printer reference.)
NULL (The *localPrinterRefPtr* parameter contains the new printer reference.)

If the **NWDPPrtAddInstalledPrinter** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00040001L NWDP_LE_FILE_NO_ACCESS_RIGHTS
0x01000004L NWDP_EC_STDIO
0x030A0001L NWDP_EC_NO_MEMORY
0x000C0004L NWDP_LE_PRT_DUPLICATE_LABEL
0x000C0013L NWDP_LE_PRT_SHORTLIST_NOT_FOUND
0x0000C0015L NWDP_LE_PRT_INSTALLR_NOT_A_USER
0x000C000AL NWDP_LE_PRT_JOB_CFG_NOT_FOUND

NCP Calls

None

See Also

NWDPPrtAddInstalledPrtWConfig
NWDPPrtCreateRefBasedOnFQN

NWDPPrtAddInstalledPrtWConfig

Adds a printer to the user's installed list (shortlists)
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtAddInstalledPrtWConfig (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        printerRef,
   NWDPAttrSetRef    jobConfigRef,
   NWDPAttrSetRef    docConfigRef,
   pnstr16           label16Ptr,
   pNWDPPrtRef       localPrinterRefPtr,
   nuint32           flags);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the reference to the printer (either controlled or public access) on the user's installed list.

jobConfigRef

(IN) Specifies the reference to the attribute set containing the job attributes of the configuration (optional).

docConfigRef

(IN) Specifies the reference to the attribute set containing the document attributes of the configuration (optional).

label16Ptr

(IN) Points to the label assigned to the installed printer (this is a Unicode string).

localPrinterRefPtr

(OUT) Points to the NWDPPrtRef, the resulting installed printer reference (optional).

flags

(IN) Specify the procedure for the library to follow for printer references.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The **NWDPPrtAddInstalledPrtWConfig** function is identical to the **NWDPPrtAddInstalledPrinter** function, except the *jobConfigRef* parameter and the *docConfigRef* parameter are implied by the *jobConfigName16Prt* parameter and must be specifically provided.

For the *flags* parameter, pass one of the following;

NWDP_PRT_FLG_NO_REF
NWDP_PRT_FLG_CONVERT_REF
NULL

For the *flags* parameter, the printer references are affected as follows:

NWDP_PRT_FLG_NO_REF (The *localPrinterRefPtr* parameter should be NULL.)
NWDP_PRT_FLG_CONVERT_REF (The *localPrinterRefPtr* parameter should be NULL. The *printerRef* parameter converts to an installed printer reference.)
NULL (The *localPrinterRefPtr* parameter contains the new printer reference.)

If the **NWDPPrtAddInstalledPrtWConfig** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00040001L NWDP_LE_FILE_NO_ACCESS_RIGHTS
0x01000004L NWDP_EC_STDIO
0x030A0001L NWDP_EC_NO_MEMORY
0x000C0004L NWDP_LE_PRT_DUPLICATE_LABEL
0x000C0013L NWDP_LE_PRT_SHORTLIST_NOT_FOUND
0x0000C0015L NWDP_LE_PRT_INSTALLR_NOT_A_USER

NCP Calls

None

None

See Also

NWDPPrtAddInstalledPrinter
NWDPPrtCreateRefBasedOnAddr
NWDPPrtCreateRefBasedOnFQN

NWDPPrtAddJobConfig

Adds a configuration to the referenced printer
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtAddJobConfig (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        printerRef,
   pnstr16           jobConfigName16Ptr,
   NWDPAttrSetRef    jobConfigRef,
   NWDPAttrSetRef    docConfigRef);
```

Parameters

accessorRef

Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the reference to the printer (either controlled or public access) on the user's installed list.

jobConfigName16Ptr

(IN) Points to the name of the configuration that is added (this is a Unicode string).

jobConfigRef

(IN) Specifies the reference to the attribute set containing the job attributes of the configuration (optional).

docConfigRef

(IN) Specifies the reference to the attribute set containing the document attributes of the configuration (optional).

Return Values

0x00000000	N_SUCCESS

0	
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the referenced printer is an NDS printer object, the configuration is added to the configuration list. If it is an installed printer with no configuration, the configuration is associated with the installed printer on the user's short list. Only one configuration is allowed per installed printer.

If the **NWDPPrtAddJobConfig** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00040001L NWDP_LE_FILE_NO_ACCESS_RIGHTS
0x01000004L NWDP_EC_STDIO
0x01000005L NWDP_EC_NDS
0x000C0007L NWDP_LE_PRT_JOB_CFG_EXISTS

NCP Calls

None

See Also

NWDPPrtRemoveJobConfig

NWDPPrtAddPrinterObjToPA

Adds an NDPS printer object to a printer agent
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtAddPrinterObjToPA (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        printerRef,
   pNWDPNSrvFQN      printerFQNPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the printer agent reference to which the NetWare Directory Services (NDS) printer object is added.

printerFQNPtr

(IN) Points to the NWDPNSrvFQN structure containing the distinguished name of the NDS printer object added to the printer agent.

Return Values

0x00000000	N_SUCCESS
0x00000001	NWDP_RC_WARNING
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPPrtAddPrinterObjToPA** function returns NWDP_RC_WARNING, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x000C0010L NWDP_LE_PRT_PO_ALREADY_ASSIGNED

If the **NWDPPrtAddPrinterObjToPA** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x01000005L NWDP_EC_NDS
0x000A000AL NWDP_LE_FQN_NOT_NDPS_PRINTER

NCP Calls

None

See Also

NWDPPrtRemovePrinterObjFromPA

NWDPPrtChangeMedia

Changes the media for a printer
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtChangeMedia (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        printerRef,
   nuint32           trayNumber,
   pNWDPNameOrOid    mediumValuePtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the printer reference.

trayNumber

(IN) Specifies the tray number.

mediumValuePtr

(IN) Points to the NWDPNameOrOid structure containing the medium value (pass NULL to set the default).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The **NWDPPrtChangeMedia** function modifies the Input Current Medium attribute of an input object in the MODB.

If the **NWDPPrtChangeMedia** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

NWDPPrtControl

Controls a printer
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtControl (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        printerRef,
   pNWDPOid          actionOidPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the printer reference.

actionOidPtr

(IN) Points to the OID indicating upcoming printer action.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The following are well-known OIDs that can be passed for the *actionOidPtr*

actionOidPtr parameter:

pNDPSVAL_DEV_CTRL_PAUSE_INPUT (*accessorRef*)
pNDPSVAL_DEV_CTRL_RESUME_INPUT (*accessorRef*)
pNDPSVAL_DEV_CTRL_PAUSE_OUTPUT (*accessorRef*)
pNDPSVAL_DEV_CTRL_RESUME_OUTPUT (*accessorRef*)
pNDPSVAL_DEV_CTRL_RESET (*accessorRef*)
pNDPSVAL_DEV_CTRL_FORM_FEED (*accessorRef*)
pNDPSVAL_DEV_CTRL_CONTINUE (*accessorRef*)

If the **NWDPPrtControl** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN

NCP Calls

None

NWDPPrtCreateMODObject

Creates an object in the Managed Object Database (MODB)
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtCreateMODObject (
   NWDPAccessorRef             accessorRef,
   NWDPPrtRef                  printerRef,
   pNWDPOid                    objectClassPtr,
   pNWDPObjectIdentification   objectIdentPtr,
   nbool                       force,
   pNWDPObjectIdentification   referenceObjectPtr,
   NWDPAttrSetRef              objectAttrSetRef,
   pNWDPAttrSetRef             resultAttrSetRefPtr,
   pNWDPASAVPRef                resultAvpRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the printer reference.

objectClassPtr

(IN) Points to the object class of the new object which is a NDPS_OC_xxxx definition in the nwdp_wko.ogh file.

objectIdentPtr

(IN) Points to the object identification of the new object.

force

(IN) Specifies that if an existing object has the same identification as specified by the *objectIdentPtr* parameter, passing N_TRUE causes that object to be replaced; otherwise, passing N_FALSE causes the error NWDP_LE_MODB_OBJ_ALREADY_EXISTS to be returned.

referenceObjectPtr

(IN) Points to the identification for an existing MODB object whose

attributes are the new object's default attributes (optional).

objectAttrSetRef

(IN) Specifies the attribute set reference which is used to create the new object (optional).

resultAttrSetRefPtr

(OUT) Points to the NWDPAttrSetRef, the resulting attribute set reference for the new object (optional).

resultAvpRefPtr

(OUT) Points to the NWDPASAVPRef, the resulting attribute and value pointer reference (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The attributes identified by the *objectAttrSetRef* parameter replace similar attributes identified by the *referenceObjectPtr* parameter.

If the **NWDPPrtCreateMODObject** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x000F0003L NWDP_LE_MODB_OBJ_ALREADY_EXISTS

NCP Calls

None

See Also

NWDPPrtDeleteMODObject

NWDPPrtCreateRefBasedOnAddr

Creates a printer reference based on a network address
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtCreateRefBasedOnAddr (
   NWDPAccessorRef   accessorRef,
   pNWDPNetAddress   netAddrPtr,
   pnstr16           paName16Ptr,
   pNWDPPrtRef       printerRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

netAddrPtr

(IN) Points to the address of the Novell Distributed Printer Services (NDPS) manager.

paName16Ptr

(IN) Points to the name of the preferred printer agent (this is a Unicode string).

printerRefPtr

(OUT) Points to the NWDPPrtRef, the resulting printer reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The printer reference created is the NWDP_PRT_REF_TYPE_CONVENIENCE type.

If the **NWDPPrtCreateRefBasedOnAddr** function returns N_FAILURE, the NWDPLibErrorMac (*accessorRef)* contains:

0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPPrtReleaseRef

NWDPPrtCreateRefBasedOnFQN

Creates a printer reference based on the Fully Qualified Name (FQN) of an NDPS printer

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtCreateRefBasedOnFQN (
   NWDPAccessorRef   accessorRef,
   pNWDPNSrvFQN      printerFqnPtr,
   pNWDPPrtRef       printerRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerFqnPtr

(IN) Points to the NWDPNSrvFQN structure containing the distinguished name of the NDPS printer object (this object resides in NDS).

printerRefPtr

(OUT) Points to the NWDPPrtRef, the resulting printer reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

Remarks

The printer reference created is the NWDP_PRT_REF_TYPE_NAMED type. It is considered a controlled-access printer.

If the **NWDPPrtCreateRefBasedOnFQN** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000005L NWDP_EC_NDS
0x030A0001L NWDP_EC_NO_MEMORY
0x000A000AL NWDP_LE_FQN_NOT_NDPS_PRINTER

NCP Calls

None

See Also

NWDPPrtReleaseRef

NWDPPrtCreateRefBasedOnLabel

Creates a printer reference based on the label of an installed printer
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtCreateRefBasedOnLabel (
   NWDPAccessorRef   accessorRef,
   pnstr16           label16Ptr,
   pNWDPPrtRef       printerRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

label16Ptr

(IN) Points to the label that uniquely identifies the installed printer (this is a Unicode string).

printerRefPtr

(OUT) Points to the NWDPPrtRef, the resulting printer reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The printer reference is the NWDP_PRT_REF_TYPE_INSTALLED type.

The printer reference is the NWDP_PRT_REF_TYPE_INSTALLED type.

If the **NWDPPrtCreateRefBasedOnLabel** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000004L NWDP_EC_STDIO
0x030A0001L NWDP_EC_NO_MEMORY
0x000C0001L NWDP_LE_PRT_NO_SUCH_LABEL

NCP Calls

None

See Also

NWDPPrtReleaseRef

NWDPPrtCreateRefBasedOnPort

Creates a printer reference based on the installed printer port
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtCreateRefBasedOnPort (
   NWDPAccessorRef   accessorRef,
   pnstr16           port16Ptr,
   pNWDPPrtRef       printerRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

port16Ptr

(IN) Points to the string containing the installed printer port (this is a Unicode string).

printerRefPtr

(OUT) Points to the NWDPPrtRef, the resulting printer reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The printer reference created is the

The printer reference created is the NWDP_PRT_REF_TYPE_INSTALLED type.

If the **NWDPPrtCreateRefBasedOnPort** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000004L NWDP_EC_STDIO
0x030A0001L NWDP_EC_NO_MEMORY
0x000C000BL NWDP_LE_PRT_NO_SUCH_PORT

NCP Calls

None

See Also

NWDPPrtReleaseRef

NWDPPrtCreateRefBasedOnPSM

Creates a printer reference based on an NDPS manager reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtCreateRefBasedOnPSM (
   NWDPAccessorRef   accessorRef,
   NWDPPsmRef        psmRef,
   pnstr16           paName16Ptr,
   pNWDPPrtRef       printerRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

psmRef

(IN) Specifies the reference to the NDPS manager.

paName16Ptr

(IN) Points to the name of the desired printer agent (this is a Unicode string).

printerRefPtr

(OUT) Points to the NWDPPrtRef, the resulting printer reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The printer reference created is the NWDP_PRT_REF_TYPE_CONVENIENCE type.

If the **NWDPPrtCreateRefBasedOnPSM** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000005L NWDP_EC_NDS
0x030A0001L NWDP_EC_NO_MEMORY
0x000B0005L NWDP_LE_NSRV_NO_ADDRESS

NCP Calls

None

See Also

NWDPPrtReleaseRef

NWDPPrtDeleteAllJobs

Deletes all jobs associated with a printer agent
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtDeleteAllJobs (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        printerRef,
   pNWDPNameOrOid    reservedPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the printer reference.

reservedPtr

(IN) Is reserved (pass NULL).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

All job objects (including retained jobs) are removed from the MODB.

If the **NWDPPrtDeleteAllJobs** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN

NCP Calls

None

See Also

NWDPPrtCreateRefBasedOnAddr
NWDPPrtCreateRefBasedOnFQN
NWDPPrtCreateRefBasedOnLabel

NWDPPrtDeleteMODObject

Deletes an object in the Managed Object Database (MODB)
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtDeleteMODObject (
   NWDPAccessorRef             accessorRef,
   NWDPPrtRef                  printerRef,
   pNWDPOid                    objectClassPtr,
   pNWDPObjectIdentification   objectIdentPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the printer reference.

objectClassPtr

(IN) Points to the object class of the new object which is a NDPS_OC_xxxx definition in the nwdp_wko.ogh file.

objectIdentPtr

(IN) Points to the object identification of the deleted object.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPPrtDeleteMODObject** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x000F0001L NWDP_LE_MODB_NO_SUCH_OBJECT

NCP Calls

None

See Also

NWDPPrtCreateMODObject

NWDPPrtDownloadInterventionCB

Presents files to be downloaded
Local Servers: N/A
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_TYPEDEF_CALLBACK (nint, NWDPPrtDownloadInterventionCB) (
     NWDPAccessorRef                 accessorRef,
     nparam                          callerDefinedParam,
     nint                            totalCallsToBeMade,
     nint                            currentCallCount,
     pnstr                           filenameToBeCopiedPtr,
     nint32                          toBeCopiedFileDate,
     nint32                          originalFileDate,
     pNWDPPrtDrvDownloadOptionEnum   currentDownloadOptionPtr,
     pNWDPPrtDrvDownloadActionEnum   actionToBeTakenPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

callerDefinedParam

(IN) Specifies a caller-defined contextual variable (optional).

totalCallsToBeMade

(IN) Specifies the number of times the **NWDPPrtDownloadInterventionCB** function is called by the **NWDPPrtDriverDownload** function (number of services plus one).

currentCallCount

(IN) Specifies the current call number.

filenameToBeCopiedPtr

(IN) Points to the name of the file to be copied.

toBeCopiedFileDate

(IN) Specifies the date of the file to be downloaded.

originalFileDate

(IN) Specifies the date of the destination file to be overwritten.

currentOptionDownloadPtr

(IN/OUT) Points to the download option that is currently in use and allows this value to be changed by the callback.

actionToBeTakenPtr

(OUT) Points to the download action to be taken.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFF	N_FAILURE

Remarks

For the *callerDefinedParam* parameter, this function uses the contextual variable to create the following:

a linked list head into which items can be added

a window handle where items can be displayed

any other data or combination of these two steps needed by the callback routine (optional)

NCP Calls

None

See Also

NWDPPrtDriverDownload

NWDPPrtDriverDownload

Downloads printer drivers for an installed printer
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtDriverDownload (
   NWDPAccessorRef                 accessorRef,
   NWDPPrtRef                      printerRef,
   NWDPResRef                      optionalResourceRef,
   pNWDPResListDrvFiles            optionalDriverIdPtr,
   NWDPPrtDrvDownloadOptionEnum    downloadOption,
   pnstr                           sourcePathOptionPtr,
   NWDPPrtDownloadInterventionCB   callback,
   nparam                           callerDefinedParam);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the reference to the installed printer which has no driver installed.

optionalResourceRef

(IN) Specifies a reference to a resource manager which has the requested drivers (optional).

optionalDriverIdPtr

(IN) Points to an NWDPResListDrvFiles structure containing the driver and .inf file information.

downloadOption

(IN) Specifies how the download occurs.

sourcePathOptionPtr

(IN) Points to the optional path for the sources, for example, A:\ (optional).

callback

(IN) Specifies the application-supplied callback function for download intervention (optional).

callerDefinedParam

(IN) Specifies a caller-defined parameter, containing application-specific data, that is passed to the callback function (optional).

Return Values

0x00000000	N_SUCCESS
0x00000001	NWDP_RC_WARNING
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *optionalResourceRef* and *optionalDriverIdPtr* parameters must both be set to valid pointers or to NULL. If NULL, the printer defined by the *printerRef* parameter is queried for the NDPS_ATT_RESOURCE_PRT_ID. The information contained in the attribute populates the NWDPResListDrvFiles structure and replaces the *optionalDriverIdPtr* parameter. The printer is also queried for the NDPS_ATT_PRT_CUR_RESOURCE_SERV attribute. The location of the current resource service creates NWDPResRef to replace the *optionalResourceRef* parameter. If this attribute does not exist, the **NWDPResCreateRefBaseOnSrsSAP** function provides the needed reference.

If the query for NDPS_ATT_RESOURCE_PRT_ID fails, make another query for the NDPS_ATT_PRINTER_DEVICE_ID. This query returns the Plug and Play identification from the P1284 port printers. (All printers do not contain the Plug and Play identification, thus, it is not the primary source of information.)

If the files are downloaded from a path containing the .inf file, the source file path is passed as a *sourcePathOptionPtr* parameter.

The *callback* and *callerDefinedParam* parameters are only used if the *downloadOption* parameter includes one of the following:

NWDP_PRT_DRV_INTERVENE_EACH_CPY
NWDP_PRT_DRV_INTERVENE_OVERWRITE

NWDP_PRT_DRV_INTERVENE_OVRW_NEW

The other values for the *downloadOption* parameter do not require a callback and their names explain the expected behavior as each file in the .inf file is copied.

The callback routine is called prior to copying, overwriting, or determining which file is more current than the source.

If the **NWDPPrtDriverDownload** function returns NWDP_RC_WARNING, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00130007L NWDP_LE_PRT_WINDOWS_RESTART_RQD

If the **NWDPPrtDriverDownload** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00100001L NWDP_LE_ATR_NOT_IN_SET
0x00130001L NWDP_LE_PRT_DRV_NONE_FOR_PLAT
0x00130002L NWDP_LE_PRT_SPCD_BROKER_NOT_FND
0x00130003L NWDP_LE_PRT_UNSUPTED_QNAME_FMT
0x00130004L NWDP_LE_PRT_NO_DEST_PATH
0x00130005L NWDP_LE_PRT_DEVICE_ID_FMT_BAD
0x00130006L NWDP_LE_PRT_NO_TMP_DEST_PATH

NCP Calls

None

NWDPPrtGetAttributeSet

Returns the attribute set of the referenced printer
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtGetAttributeSet (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        printerRef,
   NWDPOidSetRef     requestedOidSetRef,
   pNWDPAttrSetRef   resultAttrSetRefPtr,
   pNWDPASAVPRef     avpRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the printer object reference in the MODB for which the attribute set is returned.

requestedOidSetRef

(IN) Specifies the attribute set returned for the printer object (pass NULL to return all of the attributes for this object).

resultAttrSetRefPtr

(OUT) Points to the NWDPAttrSetRef, the resulting attribute set reference of the printer object.

avpRefPtr

(OUT) Points to the NWDPASAVPRef, the resulting attribute and value pointer reference (optional).

Return Values

0x00000000	N_SUCCESS

0	
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *requestedOidSetRef* parameter should contain the attribute identifier for each preferred attribute. See the nwdp_wko.ogh file for a list of the OIDs.

Call the **NWDPASReleaseRef** function to free allocated memory.

The attribute and value pointers of the *avpRefPtr* parameter are set to the first attribute and value in the printer's attribute set.

If the **NWDPPrtGetAttributeSet** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPASListAttributes
NWDPASReleaseRef
NWDPOSCreateRef
NWDPPrtCreateRefBasedOnAddr
NWDPPrtCreateRefBasedOnFQN
NWDPPrtCreateRefBasedOnLabel

NWDPPrtGetDefaultPrinter

Retrieves the current default printer for the workstation
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtGetDefaultPrinter (
   NWDPAccessorRef   accessorRef,
   pNWDPPrtRef       printerRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRefPtr

(OUT) Points to the NWDPPrtRef, the resulting printer reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The printer reference created is the NWDP_PRT_REF_TYPE_INSTALLED type.

If the **NWDPPrtGetDefaultPrinter** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000004L NWDP_EC_STDIO
0x030A0001L NWDP_EC_NO_MEMORY
0x000C0002L NWDP_LE_PRT_NO_DEFAULT

NCP Calls

None

See Also

NWDPPrtReleaseRef
NWDPPrtSetDefaultPrinter

NWDPPrtGetDriverKeyName

Retrieves the printer driver key name of an installed printer
Local Servers: nonblocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: Win
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtGetDriverKeyName (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        printerRef,
   nuint             sizeOfBuffer,
   pnstr16           driverKeyNameBuffer16Ptr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the reference to the installed printer.

sizeOfBuffer

(IN) Specifies the size in bytes of the buffer pointed to by the *driverKeyNameBuffer16Ptr* parameter.

driverKeyNameBuffer16Ptr

(OUT) Points to the buffer containing the driver key name of the installed printer (this is a Unicode string).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *driverKeyNameBuffer16Ptr* parameter points to a buffer size of NWDP_PRT_MAX_NAME_SIZE.

NCP Calls

None

See Also

NWDPPrtGetDriverName

NWDPPrtGetDriverName

Retrieves the printer driver name of the installed printer
Local Servers: nonblocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: Win
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtGetDriverName (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        printerRef,
   nuint             sizeOfBuffer,
   pnstr16           driverNameBuffer16Ptr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the reference to the installed printer.

sizeOfBuffer

(IN) Specifies the size in bytes of the buffer pointed to by the *driverNameBuffer16Ptr* parameter.

driverNameBuffer16Ptr

(OUT) Points to the buffer containing the driver name of the specified installed printer (this is a Unicode string).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *driverNameBuffer16Ptr* parameter points to a buffer size of NWDP_PRT_MAX_DRIVER_BYTES. It will contain the driver filename.

NCP Calls

None

See Also

NWDPPrtGetDriverKeyName

NWDPPrtGetFQN

Returns the Fully Qualified Name (FQN) of the printer reference
Local Servers: nonblocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtGetFQN (
   NWDPAccessorRef      accessorRef,
   NWDPPrtRef           printerRef,
   nuint                sizeOfBuffer,
   pNWDPNSrvFQNBuffer   fqnBufferPtr,
   pnuint               sizeOfResultPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the printer reference.

sizeOfBuffer

(IN) Points to the buffer size of the *fqnBufferPtr* parameter.

fqnBufferPtr

(OUT) Points to the resulting FQN (pass the address of the NWDPNSrvFQNBuffer structure).

sizeOfResultPtr

(IN/OUT) Points to the size in bytes of the data copied into the *fqnBufferPtr* parameter (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFF	NWDP_RC_INVALID_ACCESSOR

FE	
0xFFFFFFFF	N_FAILURE

Remarks

The printer reference must have an associated FQN or an error is returned.

An alternative method that saves memory is to pass a zero for the *sizeOfBuffer* parameter. The NWDP_LE_RESULT_BUFFER_TOO_SMALL error is returned with the *sizeOfResultPtr* parameter identifying the needed buffer size. Therefore, you need to allocate the amount of memory identified by the *sizeOfResultPtr* parameter, cast the buffer as a NWDPNSrvFQNBuffer structure, and pass it as the *fqnBufferPtr* parameter.

If the **NWDPPrtGetFQN** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x000A0006L NWDP_LE_RESULT_BUFFER_TOO_SMALL
0x000A0008L NWDP_LE_REF_NOT_CREATED_WITH_FQN

NCP Calls

None

See Also

NWDPPrtCreateRefBasedOnFQN
NWDPPrtCreateRefBasedOnLabel

NWDPPrtGetJobConfig

Retrieves a configuration from the referenced printer
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtGetJobConfig (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        printerRef,
   nuint             sizeOfJobConfigNameBuffer,
   pnstr16           jobConfigName16Ptr,
   pnuint            sizeOfResultNamePtr,
   pNWDPAttrSetRef   jobConfigRefPtr,
   pNWDPASAVPRef     jobAvpRefPtr,
   pNWDPAttrSetRef   docConfigRefPtr,
   pNWDPASAVPRef     docAvpRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the reference to the printer (either controlled access or installed) from which the configuration is retrieved.

sizeOfJobConfigNameBuffer

(IN) Specifies the buffer size to hold the configuration name.

jobConfigName16Ptr

(IN/OUT) Points to the name of the configuration (a Unicode string) being retrieved (optional for installed printers).

sizeOfResultNamePtr

(OUT) Points to the size in bytes of the data copied into the *jobConfigName16Ptr* parameter for installed printers (optional).

jobConfigRefPtr

(OUT) Points to the NWDPAttrSetRef, the resulting attribute set reference containing the job configuration attributes.

jobAvpRefPtr

(OUT) Points to the NWDPASAVPRef, the resulting attribute value pointer reference for the attribute set identified by the *jobConfigRefPtr* parameter (optional).

docConfigRefPtr

(OUT) Points to the NWDPAttrSetRef, the resulting attribute set reference containing the documentation configuration attributes.

docAvpRefPtr

(OUT) Points to the NWDPASAVPRef, the resulting attribute and value pointer reference for the attribute set identified by the *docConfigRefPtr* parameter (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

For installed printers, the configuration name is returned in the *jobConfigName16Ptr* parameter. For controlled-access printers, the name of the desired configuration must be specified in the *jobConfigName16Ptr* parameter.

Call the **NWDPASReleaseRef** function for each attribute set reference to free allocated memory.

If the **NWDPPrtGetJobConfig** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00040001L NWDP_LE_FILE_NO_ACCESS_RIGHTS
0x01000004L NWDP_EC_STDIO
0x01000005L NWDP_EC_NDS
0x000C000AL NWDP_LE_PRT_JOB_CFG_NOT_FOUND

NCP Calls

None

See Also

Print Service Group

NWDPPrtAddJobConfig
NWDPPrtListJobConfigs

NWDPPrtGetLabel

Retrieves the label from the referenced installed printer
Local Servers: nonblocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtGetLabel (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        printerRef,
   nuint             sizeOfBuffer,
   pnstr16           label16Ptr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the installed printer reference.

sizeOfBuffer

(IN) Specifies the buffer size pointed to by the *labelBufferPtr* parameter.

label16Ptr

(IN) Points to the buffer that receives the label of the installed printer (this is a Unicode string).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *label16Prt* parameter should point to a buffer size of NWDP_PRT_MAX_LABEL_BYTES.

NCP Calls

None

See Also

NWDPPrtCreateRefBasedOnLabel
NWDPPrtSetLabel

NWDPPrtGetMODObjectAttrSet

Returns the attribute set of the object specified in the Managed Object Database (MODB)

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtGetMODObjectAttrSet (
   NWDPAccessorRef             accessorRef,
   NWDPPrtRef                  printerRef,
   pNWDPOid                    objectClassPtr,
   pNWDPObjectIdentification   objectIdentPtr,
   NWDPOidSetRef               requestedOidSetRef,
   pNWDPAttrSetRef             resultAttrSetRefPtr,
   pNWDPASAVPRef               resultAvpRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the printer reference.

objectClassPtr

(IN) Points to the object class of the new object which is a NDPS_OC_xxxx definition in the nwdp_wko.ogh file.

objectIdentPtr

(IN) Points to the identification of the object in the MODB.

requestedOidSetRef

(IN) Specifies the attribute set returned for the object (pass NULL to return all of the attributes for this object).

resultAttrSetRefPtr

(IN) Points to the NWDPAttrSetRef, the resulting attribute set reference for the object.

resultAvpRefPtr

(IN) Points to the NWDPASAVPRef, the resulting attribute and value pointer reference (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

Call the **NWDPASReleaseRef** function to free allocated memory.

If the **NWDPPrtGetMODObjectAttrSet** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x000F0001L NWDP_LE_MODB_NO_SUCH_OBJECT

NCP Calls

None

See Also

NWDPASReleaseRef
NWDPOSCreateRef
NWDPPrtGetAttributeSet

NWDPPrtGetPort

Retrieves the current port of an installed printer
Local Servers: nonblocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtGetPort (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        printerRef,
   nuint             sizeOfBuffer,
   pnstr16           portBuffer16Ptr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the reference to the installed printer.

sizeOfBuffer

(IN) Specifies the size in bytes of the buffer pointed to by the *portBuffer16Ptr* parameter.

portBuffer16Ptr

(OUT) Points to the buffer containing the port (this is a Unicode string).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *portBuffer16Ptr* parameter should point to a buffer size of NWDP_PRT_MAX_PORT_BYTES. The expected values range from NDPS01: to NDPS100: in Unicode.

NCP Calls

None

NWDPPrtGetPrinterInfo

Returns information for the referenced printer
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtGetPrinterInfo (
   NWDPAccessorRef          accessorRef,
   NWDPPrtRef               printerRef,
   pNWDPPrtGlobalListItem   infoItemPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the reference to the printer.

infoItemPtr

(OUT) Points to the NWDPPtrGlobalListItem structure containing the printer information.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPPrtGetPrinterInfo** function returns N_FAILURE, the

If the **NWDPPrtGetPrinterInfo** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER
0x01000004L NWDP_EC_STDIO
0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

NWDPPrtGetRefType

Returns the printer type of the given reference
Local Servers: nonblocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtGetRefType (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        printerRef,
   pnuint32          refTypeFlagsPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the printer reference.

refTypeFlagsPtr

(OUT) Points to the resulting reference type.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

Installed printers (NWDP_PRT_REF_TYPE_INSTALLED) are also the NWDP_PRT_REF_TYPE_NAMED type or the

NWDP_PRT_REF_TYPE_NAMED type or the NWDP_PRT_REF_TYPE_CONVENIENCE type.

Possible values for the *refTypeFlagsPtr* parameter include one of the following:

NWDP_PRT_REF_TYPE_INSTALLED | NWDP_PRT_REF_TYPE_NAMED
NWDP_PRT_REF_TYPE_INSTALLED | NWDP_PRT_REF_TYPE_CONVENIENCE
NWDP_PRT_REF_TYPE_NAMED
NWDP_PRT_REF_TYPE_CONVENIENCE

NCP Calls

None

NWDPPrtGetStatus

Retrieves the status of the referenced printer
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtGetStatus (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        printerRef,
   pNWDPAttrSetRef   resultAttrSetRefPtr,
   pNWDPASAVPRef     avpRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the printer reference from which the status is queried.

resultAttrSetRefPtr

(OUT) Points to the NWDPAttrSetRef, the resulting attribute set reference from the printer status attributes.

avpRefPtr

(OUT) Points to the NWDPASAVPRef, the resulting attribute and value pointer reference (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The attributes returned in the *resultAttrSetRefPtr* parameter are printer state, printer state reasons, and printer enabled. The attribute identifiers (see the nwdp_wko.ogh file) are the following:

pNDPSATT_PRINTER_STATE *(accessorRef)*
pNDPSATT_PRINTER_STATE_REASONS *(accessorRef)*
pNDPSATT_PRT_STATE_SEVERITY *(accessorRef)*
pNDPSATT_PRINTER_ENABLED *(accessorRef)*

These attributes are queried from the MODB.

A List Object Attributes (LOA) does not fail if a particular attribute is not found. Call the **NWDPASSetAVPByAttributeId** function to determine which attributes, if any, are returned.

Call the **NWDPASReleaseRef** function to free allocated memory

If the **NWDPrtGetStatus** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPASReleaseRef
NWDPASSetAVPByAttributeId

NWDPPrtGlobalListCallback

Returns information on job objects through the **NWDPPrtListGlobalPrinters** function

Local Servers: N/A

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_TYPEDEF_CALLBACK (nint, NWDPPrtGlobalListCallback) (
   NWDPAccessorRef          accessorRef,
   nparam                   listCallbackParam2,
   nint                     totalCallsToBeMade,
   nint                     currentCallCount,
   pNWDPPrtGlobalListItem   itemReceivedPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

listCallbackParam2

(IN) Specifies a user-defined contextual variable used by the **NWDPPrtGlobalListCallback** function.

totalCallsToBeMade

(IN) Specifies the number of times to call the **NWDPPrtGlobalListCallback** function (the number of printers plus one).

currentCallCount

(IN) Specifies the current call number.

itemReceivedPtr

(IN) Points to the NWDPPrtGlobalListItem structure.

Return Values

0x00000000	N_SUCCESS

0	
0xFFFFFFFF	N_FAILURE
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR

Remarks

For the *listCallbackParam2* parameter, this function uses the contextual variable to create the following:

a linked list head into which items can be added

a window handle where items can be displayed

any other data or combination of these two steps needed by the callback routine (optional)

For the *itemReceivedPtr* parameter, NULL is passed on the last call to the **NWDPPrtGlobalListCallback** function.

NCP Calls

None

See Also

NWDPPrtListGlobalPrinters

NWDPPrtInstalledListCallback

Returns information through the **NWDPPrtListInstalledPrinters** function
Local Servers: N/A
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_TYPEDEF_CALLBACK (nint, NWDPPrtInstalledListCallback) (
   NWDPAccessorRef             accessorRef,
   param                       listCallbackParam2,
   nint                        totalCallsToBeMade,
   nint                        currentCallCount,
   pNWDPPrtInstalledListItem   itemReceivedPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

listCallbackParam2

(IN) Specifies a user-defined contextual variable.

totalCallsToBeMade

(IN) Specifies the number of times to call the **NWDPPrtInstalledListCallback** function (the number of captured printers plus one).

currentCallCount

(IN) Specifies the current call number.

itemReceivedPtr

(IN) Points to the NWDPPrtInstalledListItem structure.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFF	N_FAILURE

FF	
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR

Remarks

For the *listCallbackParam2* parameter, this function uses the contextual variable to create the following:

a linked list head into which items can be added

a window handle where items can be displayed

any other data or combination of these two steps needed by the callback routine (optional)

For the *itemReceivedPtr* parameter, NULL is passed on the last call to the **NWDPPrtInstalledListCallback** function.

NCP Calls

None

See Also

NWDPPrtListInstalledPrinters

NWDPPrtJobConfigListCallback

Returns information on job objects through the **NWDPPrtListJobConfigs** function

Local Servers: N/A

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_TYPEDEF_CALLBACK (nint, NWDPPrtJobConfigListCallback) (
   NWDPAccessorRef   accessorRef,
   nparam            listCallbackParam2,
   nint              totalCallsToBeMade,
   nint              currentCallCount,
   pnstr16           itemReceived16Ptr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

listCallbackParam2

(IN) Specifies a user-defined contextual variable.

totalCallsToBeMade

(IN) Specifies the number of times to call the **NWDPPrtJobConfigListCallback** function (the number of job configurations plus one).

currentCallCount

(IN) Specifies the current call number.

itemReceived16Ptr

(IN) Points to the job configuration name.

Return Values

0x00000000	N_SUCCESS

0xFFFFFFFF	N_FAILURE
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR

Remarks

For the *listCallbackParam2* parameter, this function uses the contextual variable to create the following:

a linked list head into which items can be added

a window handle where items can be displayed

any other data or combination of these two steps needed by the callback routine (optional)

For the *itemReceived16Ptr* parameter, NULL is passed on the last call to the **NWDPPrtJobConfigListCallback** function.

NCP Calls

None

See Also

NWDPPrtListJobConfigs

NWDPPrtJobCopy

Copies a job within the same printer agent or between two printer agents
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtJobCopy (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        destPrinterRef,
   NWDPPrtRef        srcPrinterRef,
   nuint32           jobId,
   pNWDPJobRef       destJobRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

destPrinterRef

(IN) Specifies the printer reference that receives the job (pass NULL if the copy is within the same device).

srcPrinterRef

(IN) Specifies the reference to the printer containing the copied job.

jobId

(IN) Specifies the identifier of the copied job.

destJobRefPtr

(OUT) Points to the NWDPJobRef, the resulting job reference (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFF	NWDP_RC_INVALID_ACCESSOR

FE	
0xFFFFFFFF	N_FAILURE

Remarks

If the source and destination printers are controlled-access printers (represented in NDS), they must reside within the same NDS directory tree.

If the **NWDPPrtJobCopy** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00090005L NWDP_LE_JOB_NO_SUCH_JOB
0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x030A0001L NWDP_EC_NO_MEMORY
0x000C000DL NWDP_LE_PRT_DEST_TREE_DIFFERS

NCP Calls

None

See Also

NWDPJobCopy
NWDPJobMove
NWDPPrtJobMove

NWDPPrtJobListCallback

Lists jobs
Local Servers: N/A
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_TYPEDEF_CALLBACK (nint, NWDPPrtJobListCallback) (
   NWDPAccessorRef       accessorRef,
   nparam                callerDefinedParam,
   nint                  totalCallsToBeMade,
   nint                  currentCallCount,
   pNWDPPrtJobListItem   itemReceivedPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

callerDefinedParam

(IN) Specifies a caller-defined contextual variable.

totalCallsToBeMade

(IN) Specifies the number of times the **NWDPPrtListJobs** function calls the **NWDPPrtJobListCallback** function (number of jobs plus one).

currentCallCount

(IN) Specifies the current call number.

itemReceivedPtr

(IN) Points to the NWDPPrtJobListItem structure containing the job information.

Return Values

0x00000000	N_SUCCESS

0xFFFFFFFF	N_FAILURE

Remarks

For the *callerDefinedParam* parameter, this function uses the contextual variable to create the following:

a linked list head into which items can be added

a window handle where items can be displayed

any other data or combination of these two steps needed by the callback routine (optional)

For the *totalCallsToBeMade* parameter, the *itemReceivedPtr* parameter is NULL for the last callback.

NCP Calls

None

See Also

NWDPPrtListJobs

NWDPPrtJobMove

Moves a job to another printer
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtJobMove (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        destPrinterRef,
   NWDPPrtRef        srcPrinterRef,
   nuint32           jobId,
   pNWDPJobRef       destJobRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

destPrinterRef

(IN) Specifies the reference to the destination printer.

srcPrinterRef

(IN) Specifies the reference to the printer containing the moved job.

jobId

(IN) Specifies the identifier of the moved job.

destJobRefPtr

(IN) Points to the NWDPJobRef, the resulting job reference (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFF	N_FAILURE

FF	

Remarks

If the source and destination printers are controlled-access printers (represented in NDS), they must reside within the same NDS directory tree.

If the **NWDPPrtJobMove** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains:

0x00090005L NWDP_LE_JOB_NO_SUCH_JOB
0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x030A0001L NWDP_EC_NO_MEMORY
0x000C000DL NWDP_LE_PRT_DEST_TREE_DIFFERS

NCP Calls

None

See Also

NWDPJobCopy
NWDPJobMove
NWDPPrtJobCopy

NWDPPrtListGlobalPrinters

Retrieves all available NetWare printers
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtListGlobalPrinters (
   NWDPAccessorRef             accessorRef,
   NWDPFilterRef               filterRef,
   NWDPPrtRefTypeEnum          printerTypeFilter,
   NWDPPrtGlobalListCallback   listCallBack,
   nparam                       callerDefinedParam);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

filterRef

(IN) Specifies the NWDPFilterRef filter used to determine the returned printers.

printerTypeFilter

(IN) Specifies the type of printers to list.

listCallBack

(IN) Specifies the **NWDPPrtGlobalListCallback** function receiving the resulting printers.

callerDefinedParam

(IN) Specifies a user-defined parameter passed to the **NWDPPrtGlobalListCallback** function containing application-specific data.

Return Values

0x00000000	N_SUCCESS

0	
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The *printerTypeFilter* parameter can have the following values:

2 NWDP_PRT_REF_TYPE_NAMED
4 NWDP_PRT_REF_TYPE_CONVENIENCE
7 NWDP_PRT_REF_TYPE_ANY

The *callerDefinedParam* parameter can contain the box handle where the list is inserted.

Also, the *callerDefinedParam* parameter is passed to the *listCallBack* parameter as the second parameter.

If the **NWDPPrtListGlobalPrinters** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER
0x01000005L NWDP_EC_NDS
0x030A0001L NWDP_EC_NO_MEMORY
0x000A0003L NWDP_LE_CANCEL_LIST
0x000B0001L NWDP_LE_NSRV_NO_CONTEXT_IN_USE

NCP Calls

None

See Also

NWDPPrtGlobalListCallback

NWDPPrtListInstalledPrinters

Lists the user's installed devices
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtListInstalledPrinters (
   NWDPAccessorRef                accessorRef,
   NWDPPrtInstalledListCallback   listCallBack,
   nparam                          callerDefinedParam);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

listCallBack

(IN) Specifies the **NWDPPrtInstalledListCallback** function receiving the resulting devices.

callerDefinedParam

(IN) Specifies a user-defined parameter passed to the **NWDPPrtInstalledListCallback** function containing application-specific data.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

Remarks

The *callerDefinedParam* parameter can contain the box handle where the list is inserted.

Also, the *callerDefinedParam* parameter is passed to the *listCallBack* parameter as the second parameter.

If the **NWDPPrtListInstalledPrinters** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER
0x01000004L NWDP_EC_STDIO
0x000A0003L NWDP_LE_CANCEL_LIST

NCP Calls

None

See Also

NWDPPrtInstalledListCallback

NWDPPrtListJobConfigs

Lists the configuration(s) associated with the referenced printer
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtListJobConfigs (
   NWDPAccessorRef                accessorRef,
   NWDPPrtRef                     printerRef,
   NWDPPrtJobConfigListCallback   listCallBack,
   nparam                          callerDefinedParam);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the NWDPPrtRef containing the printer for which configurations are listed.

listCallBack

(IN) Specifies the **NWDPPrtJobConfigListCallback** function containing the resulting devices.

callerDefinedParam

(IN) Specifies a user-defined parameter passed to the **NWDPPrtJobConfigListCallback** function containing application-specific data.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFF	N_FAILURE

0xFFFFFFFF	N_FAILURE

Remarks

The *callerDefinedParam* parameter can contain the box handle where the list is inserted.

Also, the *callerDefinedParam* parameter is passed to the *listCallBack* parameter as the second parameter.

If the**NWDPPrtListInstalledPrinters** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER
0x01000004L NWDP_EC_STDIO
0x000A0003L NWDP_LE_CANCEL_LIST

NCP Calls

None

See Also

NWDPPrtJobConfigListCallback

NWDPPrtListJobs

Lists the jobs associated with the specified printer
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtListJobs (
   NWDPAccessorRef          accessorRef,
   NWDPPrtRef               printerRef,
   NWDPPrtJobTypeEnum       jobType,
   NWDPFilterRef            filterRef,
   NWDPPrtJobListCallback   listCallback,
   nparam                   callerDefinedParam);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the printer reference.

jobType

(IN) Specifies the types of jobs to list (NWDP_PRT_SCHEDULED_JOBS or NWDP_PRT_RETAINED_JOBS).

filterRef

(IN) Specifies a reference to a constructed filter which determines the jobs returned (optional).

listCallback

(IN) Specifies the application-supplied callback function which is called by the **NWDPPrtListJobs** function for each job.

callerDefinedParam

(IN) Specifies a caller-defined parameter, containing an application-specific data, that is passed to the callback function (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPPrtListJobs** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x030A0001L NWDP_EC_NO_MEMORY
0x000A0005L NWDP_LE_UNKNOWN_CALLBACK_ERR

NCP Calls

None

See Also

NWDPPrtCreateRefBasedOnAddr
NWDPPrtCreateRefBasedOnFQN
NWDPPrtCreateRefBasedOnLabel

NWDPPrtListMODObjects

Lists the specified Managed Object Database (MODB) objects and their respective attributes

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtListMODObjects (
   NWDPAccessorRef               accessorRef,
   NWDPPrtRef                    printerRef,
   pNWDPOid                      objectClassPtr,
   pNWDPASSelector               selectorPtr,
   NWDPOidSetRef                 requestedOidSetRef,
   NWDPASMODObjectListCallback   listCallback,
   nparam                         callerDefinedParam);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the printer reference.

objectClassPtr

(IN) Points to the object class of the new object which is a NDPS_OC_xxxx definition in the nwdp_wko.ogh file.

selectorPtr

(IN) Points to the NWDPASSelector structure that filters the selection of objects (optional).

requestedOidSetRef

(IN) Specifies the set of attributes returned for each object (pass NULL to return all of the attributes for each object).

listCallBack

(IN) Specifies the application-supplied callback function called for each object by the **NWDPPrtListMODObjects** function.

callerDefinedParam

(IN) Specifies a caller-defined parameter, containing application-specific data, that is passed to the callback function (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPPrtListMODObjects** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x030A0001L NWDP_EC_NO_MEMORY
0x000A0005L NWDP_LE_UNKNOWN_CALLBACK_ERR

NCP Calls

None

See Also

NWDPOSCreateRef

NWDPPrtModifyAttrs

Modifies the attributes of a printer object in the Managed Object Database (MODB)

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtModifyAttrs (
   NWDPAccessorRef            accessorRef,
   NWDPPrtRef                 printerRef,
   NWDPAttrSetRef             modifyRef,
   NWDPASModifyOperatorEnum   prtOperator);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the reference to the MODB printer object that has modified attributes.

modifyRef

(IN) Specifies the reference to the attribute set that modifies the printer object.

prtOperator

(IN) Specifies the modify operator that is applied to each attribute in the set identified by the *modifyRef* parameter.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFF	N_FAILURE

0xFFFFFFFF	N_FAILURE

Remarks

Each attribute in the attribute set specified by the *modifyRef* parameter has a modify operator assigned to it. This modify operator defines the modification to occur for the attribute.You can set the value for each attribute using the *qualifierPtr* parameter of the **NWDPASAddAttribute** function. Or, you can overwrite the entire attribute set value by passing a non-NULL (other than NWDP_MODIFY_OP_NULL) value for the *prtOperator* parameter.

If the **NWDPPrtModifyAttrs** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x000F0001L NWDP_LE_MODB_NO_SUCH_OBJECT

NCP Calls

None

See Also

NWDPASAddAttribute
NWDPASCreateRef
NWDPJobModifyAttrs

NWDPPrtModifyMODObjectAttrs

Modifies the attributes to an object in the MODB (Managed Object Database)

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtModifyMODObjectAttrs (
   NWDPAccessorRef             accessorRef,
   NWDPPrtRef                  printerRef,
   pNWDPOid                    objectClassPtr,
   pNWDPObjectIdentification   objectIdentPtr,
   NWDPAttrSetRef              modifyRef,
   NWDPASModifyOperatorEnum     modifyOperator);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the printer reference.

objectClassPtr

(IN) Points to the object class of the new object which is a NDPS_OC_xxxx definition in the nwdp_wko.ogh file.

objectIdentPtr

(IN) Points to the object identification of the modified object.

modifyRef

(IN) Specifies the reference to the attribute set which modifies the object.

prtOperator

(IN) Specifies the modify operator that is applied to each attribute in the set the *modifyRef* parameter identifies.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

Each attribute in the attribute set specified by the *modifyRef* parameter has a modify operator assigned to it. This modify operator defines the modification to occur for the attribute. You can set the value for each attribute by using the *qualifierPtr* parameter of the **NWDPASAddAttribute**function. Or, you can overwrite the entire attribute set value by passing a non-NULL (other than NWDP_MODIFY_OP_NULL) value for the *modifyOperator* parameter.

If the **NWDPPrtModifyMODObjectAttrs** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x000F0001L NWDP_LE_MODB_NO_SUCH_OBJECT

NCP Calls

None

See Also

NWDPASAddAttribute
NWDPASCreateRef
NWDPPrtCreateRefBasedOnAddr
NWDPPrtCreateRefBasedOnFQN
NWDPPrtCreateRefBasedOnLabel

NWDPPrtReleaseRef

Decrements the usage count of a printer reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtReleaseRef (
   NWDPAccessorRef   accessorRef,
   pNWDPPrtRef       printerRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRefPtr

(IN) Points to the released printer reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

When the usage count reaches zero, the reference is destroyed and the associated memory is freed. The document and job references of the printer reference are also destroyed.

If the **NWDPPrtReleaseRef** function returns N_FAILURE, NWDPLibErrorMac *(accessorRef)* contains the following:

NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER

NCP Calls

None

See Also

NWDPPrtAddInstalledPrinter
NWDPPrtAddInstalledPrtWConfig
NWDPPrtCreateRefBasedOnAddr
NWDPPrtCreateRefBasedOnFQN
NWDPPrtCreateRefBasedOnLabel
NWDPPrtCreateRefBasedOnPort
NWDPPrtCreateRefBasedOnPSM
NWDPPrtGetDefaultPrinter
NWDPPrtUseRef

NWDPPrtRemoveInstalledPrinter

Removes a printer from the installed printer list (shortlist) and destroys the printer reference

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtRemoveInstalledPrinter (
   NWDPAccessorRef   accessorRef,
   pNWDPPrtRef       printerRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRefPtr

(IN) Points to the reference of the installed printer that is removed.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The printer reference must be of the NWDP_PRT_REF_TYPE_INSTALLED type.

If the **NWDPPrtRemoveInstalledPrinter** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000004L NWDP_EC_STDIO
0x000C0013L NWDP_LE_PRT_SHORTLIST_NOT_FOUND

NCP Calls

None

See Also

NWDPPrtAddInstalledPrinter
NWDPPrtAddInstalledPrtWConfig

NWDPPrtRemoveJobConfig

Removes a configuration from the referenced printer
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtRemoveJobConfig (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        printerRef,
   pnstr16           jobConfigName16Ptr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the NWDPPrtRef containing an installed or named printer reference.

jobConfigName16Ptr

(IN) Points to the configuration name to be removed (optional for installed printers).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

Installed printers only have one configuration stored. This configuration is removed by passing NULL for the *jobConfigName16Ptr* parameter.

NCP Calls

None

NWDPPrtRemovePrinterObjFromPA

Removes an NDPS printer object from a printer agent
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtRemovePrinterObjFromPA (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        printerRef,
   pNWDPNSrvFQN      printerFqnPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the reference to the printer agent from which the NDPS printer object is removed.

printerFqnPtr

(IN) Points to the NWDPNSrvFQN structure containing the NDS printer object distinguished name that is removed from the printer agent.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

Remarks

If the **NWDPPrtRemovePrinterObjFromPA** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x01000005L NWDP_EC_NDS
0x000A000AL NWDP_LE_FQN_NOT_NDPS_PRINTER
0x000C0011L NWDP_LE_PRT_PO_NOT_ASSIGNED

NCP Calls

None

See Also

NWDPPrtAddPrinterObjToPA

NWDPPrtRenameJobConfig

Renames a configuration of a referenced printer
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtRenameJobConfig (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        printerRef,
   pnstr16           oldJobConfigName16Ptr,
   pnstr16           newJobConfigName16Ptr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Points to the reference of the installed or named printer.

oldJobConfigName16Ptr

(IN) Points to the old job configname.

newJobConfigName16Ptr

(IN) Points to the new job configname.

Return Values

0x00000000	N_SUCCESS

Remarks

The printer's administrator limits and/or locks certain values by providing the MOD printer object as a subordinate object class of NDPS_OC_JOB_LIMITS and/or NDPS_OC_DOC_LIMITS. This routine

NDPS_OC_JOB_LIMITS and/or NDPS_OC_DOC_LIMITS. This routine contacts the printer, retrieves the attribute set, and returns the limits storing the attribute address. The attribute set corresponds to either the NDPS_OC_JOB_LIMITS and/or NDPS_OC_DOC_LIMITS classes (which require the *printerRef* and *limitObjectClassOidPrt* parameters).

This limiting or locking of certain values omits the previous set, if the printer reference and the *limitObjectClassOidPrt* parameter are both set to NULL and the *limitsAttrSetRefPtr* parameter points to a non-NULL attribute set reference. Despite the limits pointer information, the limits from the *oldJobConfigName16Ptr* parameter are applied to the attribute set in the *newJobConfigName16Ptr* parameter.

NCP Calls

None

NWDPPrtResyncWithNameService

Tells printer to synchronize the MOD object attributes with the NDS name service

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtResyncWithNameService (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        printerRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Points to the reference of the installed or named printer.

Return Values

0x00000000	N_SUCCESS

Remarks

Before calling the **NWDPPrtResyncWithNameService** function, the NDS attributes should be modified.

The printer's administrator limits and/or locks certain values by providing the MOD printer object as a subordinate object class of NDPS_OC_JOB_LIMITS and/or NDPS_OC_DOC_LIMITS. This routine contacts the printer, retrieves the attribute set, and returns the limits storing the attribute address. The attribute set corresponds to either the NDPS_OC_JOB_LIMITS and/or NDPS_OC_DOC_LIMITS classes

(which require the *printerRef* and *limitObjectClassOidPrt* parameters).

This limiting or locking of certain values omits the previous set, if the printer reference and the*limitsObjectClassOidPrt* parameter are both set to NULL and the *limitsAttrSetRefPtr* parameter points to a non-NULL attribute set reference.

NCP Calls

None

NWDPPrtSetDefaultPrinter

Sets the current default printer for the workstation
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtSetDefaultPrinter (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        printerRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the reference to the printer that is set as the default.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The printer reference must be of the NWDP_PRT_REF_TYPE_INSTALLED type.

If the **NWDPPrtSetDefaultPrinter** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000004L NWDP_EC_STDIO
0x000C0013L NWDP_LE_PRT_SHORTLIST_NOT_FOUND

NCP Calls

None

See Also

NWDPPrtGetDefaultPrinter

NWDPPrtSetLabel

Changes the label of the referenced installed printer
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtSetLabel (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        printerRef,
   pnstr16           label16Ptr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the reference to the installed printer.

label16Ptr

(IN) Points to the new label for the referenced printer (this is a Unicode string).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPPrtSetLabel** function returns N_FAILURE, the

If the **NWDPPrtSetLabel** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000004L NWDP_EC_STDIO
0x000C0004L NWDP_LE_PRT_DUPLICATE_LABEL
0x000C0013L NWDP_LE_PRT_SHORTLIST_NOT_FOUND

NCP Calls

None

See Also

NWDPPrtCreateRefBasedOnLabel
NWDPPrtGetLabel

NWDPPrtShutdownPA

Shuts down a printer agent
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtShutdownPA (
   NWDPAccessorRef    accessorRef,
   NWDPPrtRef         printerRef,
   pNWDPNameOrOid     reservedPtr,
   NWDPShutdownType   shutdownType);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the printer agent reference that is shutdown.

reservedPtr

(IN) Is reserved (pass NULL).

shutdownType

(IN) Specifies the shutdown type.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

For the *shutdownType* parameter, pass one of the following:

NWDP_SHUTDOWN_DO_CURRENT_JOBS
NWDP_SHUTDOWN_IMMEDIATE
NWDP_SHUTDOWN_DO_PENDING_JOBS

If the **NWDPPrtShutdownPA** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN

NCP Calls

None

See Also

NWDPPrtStartupPA

NWDPPrtStartupPA

Starts a printer agent
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtStartupPA (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        printerRef,
   pNWDPNameOrOid    reservedPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the printer agent reference which is started.

reservedPtr

(IN) Is reserved (pass NULL).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPPrtStartupPA** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN

NCP Calls

None

See Also

NWDPPrtShutdownPA

NWDPPrtSyncInstalledPrinters

Causes the installed printers list and the Windows registry to be resynchronized

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtSyncInstalledPrinters (
   NWDPAccessorRef   accessorRef,
   nbool             osRegistryTakesPrecedence);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

osRegistryTakesPrecedence

(IN) Specifies a boolean.

Return Values

0x00000000	N_SUCCESS

Remarks

The **NWDPPrtSyncInstalledPrinters** function requires Windows 3.1 or Windows 95. If the *osRegistryTakesPrecedence* parameter is N_TRUE, changes to the WIN.INI or Registry entries are reflected in the installed printer list; otherwise, list changes1 are reflected in the WIN.INI and/or Registry.

The printer's administrator limits and/or locks certain values by providing the MOD printer object as a subordinate object class(es) of NDPS_OC_JOB_LIMITS and/or NDPS_OC_DOC_LIMITS. This routine

contacts the printer, retrieves the attribute set, and returns the limits storing the attribute address. The attribute set corresponds to either the NDPS_OC_JOB_LIMITS and/or NDPS_OC_DOC_LIMITS classes (which require the *printerRef* and *limitObjectClassOidPrt* parameters).

This limiting or locking of certain values omits the previous set, if the printer reference and the*limitsObjectClassOidPrt* parameter are both set to NULL and the *limitsAttrSetRefPtr* parameter points to a non-NULL attribute set reference.

NCP Calls

None

NWDPPrtUseRef

Increments the usage count of the printer reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtUseRef (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        printerRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the printer reference which has an incremented usage count.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPPrtUseRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER

NCP Calls

None

See Also

NWDPPrtAddInstalledPrinter
NWDPPrtAddInstalledPrtWConfig
NWDPPrtCreateRefBasedOnAddr
NWDPPrtCreateRefBasedOnFQN
NWDPPrtCreateRefBasedOnLabel
NWDPPrtCreateRefBasedOnPort
NWDPPrtCreateRefBasedOnPSM
NWDPPrtGetDefaultPrinter
NWDPPrtReleaseRef

NWDPPrtValidateRef

Validates a printer reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_prt.h>

N_EXTERN_LIBRARY (nint) NWDPPrtValidateRef (
   NWDPAccessorRef   accessorRef,
   NWDPPrtRef        printerRef,);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

printerRef

(IN) Specifies the printer reference to validate.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The **NWDPPrtValidateRef** function verifies that the printer reference is associated with the *accessorRef* parameter when the application has received it from an untrusted party.

If the **NWDPPrtValidateRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER

NCP Calls

None

NWDPPsmAddPA

Adds a printer agent to the NDPS Manager
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_psm.h>

N_EXTERN_LIBRARY (nint) NWDPPsmAddPA (
   NWDPAccessorRef   accessorRef,
   NWDPPsmRef        psmRef,
   pnstr16           paName16Ptr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

psmRef

(IN) Specifies the reference to the NDPS manager with the added the printer agent.

paName16Ptr

(IN) Points to the name of the printer agent that is added (this is a Unicode string).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPPsmAddPA** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x000F0003L NWDP_LE_MODB_OBJ_ALREADY_EXISTS

NCP Calls

None

See Also

NWDPPsmRemovePA

NWDPPsmAddPAAndAssignPrinterObj

Adds a printer agent to the NDPS Manager and assigns a printer object
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_psm.h>

N_EXTERN_LIBRARY (nint) NWDPPsmAddPA (
   NWDPAccessorRef   accessorRef,
   NWDPPsmRef        psmRef,
   pNWDPNSrvFQN      printerFqnPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

psmRef

(IN) Specifies the reference to the NDPS manager with the added printer agent.

printerFqnPtr

(IN) Points to the NWDPNSrvFQN structure containing the distinguished name of the printer object that is added to the new printer agent.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

Remarks

The new printer agent name is identical to the printer object name. If that particular name is already in use, a _2, _3, and so forth is appended to the name of the new printer agent.

If the **NWDPPsmAddPAAndAssignPrinterObj** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x01000005L NWDP_EC_NDS
0x000A0008L NWDP_LE_REF_NOT_CREATED_WITH_FQN
0x000A000AL NWDP_LE_FQN_NOT_NDPS_PRINTER

NCP Calls

None

See Also

NWDPPrtAddPrinterObjToPA
NWDPPsmAddPA

NWDPPsmCancelShutdownPSM

Cancels a previous request to shutdown the NDPS Manager
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_psm.h>

N_EXTERN_LIBRARY (nint) NWDPPsmCancelShutdownPSM (
   NWDPAccessorRef   accessorRef,
   NWDPPsmRef        psmRef,
   pNWDPNameOrOid    reservedPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

psmRef

(IN) Specifies the reference to the NDPS Manager with the canceled shutdown.

reservedPtr

(IN) Is reserved (pass NULL).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPPsmCancelShutdownPSM** function returns N_FAILURE,

If the **NWDPPsmCancelShutdownPSM** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN

NCP Calls

None

See Also

NWDPPsmShutdownPSM

NWDPPsmCreateMODObject

Creates an object in the MODB (Managed Object Database)
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_psm.h>

N_EXTERN_LIBRARY (nint) NWDPPsmCreateMODObject (
     NWDPAccessorRef             accessorRef,
     NWDPPsmRef                  psmRef,
     pNWDPOid                    objectClassPtr,
     pNWDPObjectIdentification   objectIdentPtr,
     nbool                       force,
     pNWDPObjectIdentification   referenceObjectPtr,
     NWDPAttrSetRef              objectAttrSetRef,
     pNWDPAttrSetRef             resultAttrSetRefPtr,
     pNWDPASAVPRef               resultAvpRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

psmRef

(IN) Specifies the reference to the NDPS Manager which manages the new object.

objectClassPtr

(IN) Points to the object class of the new object which is a NDPS_OC_xxxx definition in the nwdp_wko.ogh file.

objectIdentPtr

(IN) Points to the object identification for the new object.

force

(IN) Specifies whether or not to replace an existing object with the new object if both objects have the same identification.

referenceObjectPtr

(IN) Points to the identification for an existing MODB object whose attributes are the default attributes for the new object (optional).

objectAttrSetRef

(IN) Specifies the reference to the attribute set which is used to create the new object (optional).

resultAttrSetRefPtr

(OUT) Points to the NWDPAttrSetRef, the resulting attribute set reference for the new object (optional).

resultAvpRefPtr

(OUT) Points to the NWDPASAVPRef, the resulting attribute and value pointer reference (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If an existing object has the same identification as the *objectIdentPtr* parameter, passing N_TRUE for the *force* parameter causes that object to be replaced with the new one; otherwise, passing N_FALSE causes the error NWDP_LE_MODB_OBJ_ALREADY_EXISTS.

The attributes identified by the *objectAttrSetRef* parameter replaces attributes identified by the *referenceObjectPtr* parameter.

If the **NWDPPsmCreateMODObject** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x000F0003L NWDP_LE_MODB_OBJ_ALREADY_EXISTS

NCP Calls

None

See Also

NWDPPsmDeleteMODObject

NWDPPsmCreateRefBasedOnAddr

Creates a NDPS Manager reference based on a network address
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_psm.h>

N_EXTERN_LIBRARY (nint) NWDPPsmCreateRefBasedOnAddr (
   NWDPAccessorRef   accessorRef,
   pNWDPNetAddress   netAddrPtr,
   pNWDPPsmRef       psmRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

netAddrPtr

(IN) Points to the address of the preferred NDPS Manager.

psmRefPtr

(OUT) Points to the NWDPPsmRef, the resulting NDPS Manager reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPPsmCreateRefBasedOnAddr** function returns N_FAILURE,

If the **NWDPPsmCreateRefBasedOnAddr** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPPsmDestroyRef

NWDPPsmCreateRefBasedOnFQN

Creates a NDPS Manager reference based on its Fully Qualified Name (FQN)

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_psm.h>

N_EXTERN_LIBRARY (nint) NWDPPsmCreateRefBasedOnFQN (
   NWDPAccessorRef   accessorRef,
   pNWDPNSrvFQN      psmFqnPtr,
   pNWDPPsmRef       psmRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

psmFqnPtr

(IN) Points to the NWDPNSrvFQN structure containing the distinguished name of the NDPS Manager object.

psmRefPtr

(OUT) Points to the NWDPPsmRef, the resulting NDPS Manager reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

Remarks

If the **NWDPPsmCreateRefBasedOnFQN** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000005L NWDP_EC_NDS
0x030A0001L NWDP_EC_NO_MEMORY
0x000A0009L NWDP_LE_FQN_NOT_NDPS_MANAGER

NCP Calls

None

See Also

NWDPPsmDestroyRef

NWDPPsmDeleteMODObject

Deletes an object in the MODB (Managed Object Database)
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_psm.h>

N_EXTERN_LIBRARY (nint) NWDPPsmDeleteMODObject (
   NWDPAccessorRef             accessorRef,
   NWDPPsmRef                  psmRef,
   pNWDPOid                    objectClassPtr,
   pNWDPObjectIdentification   objectIdentPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

psmRef

(IN) Specifies the reference to the NDPS Manager.

objectClassPtr

(IN) Points to the object class of the new object which is a NDPS_OC_xxxx definition in the nwdp_wko.ogh file.

objectIdentPtr

(IN) Points to the object identification of the object that is deleted.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPPsmDeleteMODObject** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NWDP_ERROR_RETURN
0x000F0001L NWDP_LE_MODB_NO_SUCH_OBJECT

NCP Calls

None

See Also

NWDPPsmCreateMODObject

NWDPPsmDestroyRef

Destroys the NDPS Manager reference and frees all associated memory
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_psm.h>

N_EXTERN_LIBRARY (nint) NWDPPsmDestroyRef (
   NWDPAccessorRef   accessorRef,
   pNWDPPsmRef       psmRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

psmRefPtr

(IN) Points to the NDPS Manager reference that is destroyed.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPPsmDestroyRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER

NCP Calls

None

See Also

NWDPPsmCreateRefBasedOnAddr
NWDPPsmCreateRefBasedOnFQN

NWDPPsmGetConnectionStatus

Checks the status of the current connection to the NDPS Manager; otherwise, it creates a new connection

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_psm.h>

N_EXTERN_LIBRARY (nint) NWDPPsmGetConnectionStatus (
   NWDPAccessorRef   accessorRef,
   NWDPPsmRef        psmRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

psmRef

(IN) Specifies the reference to the NDPS Manager.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

A bind occurs if the *psmRef* parameter is not bound to the NDPS Manager. If the *psmRef* parameter is already bound, the connection is checked. If the connection is lost, NWDP_LE_PSM_CONNECTION_LOST is returned.

If the **NWDPPsmGetConnectionStatus** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00030001L NWDP_LE_PSM_CONNECTION_LOST
0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x05170001L NWDP_EC_PSM_BOUND_TO_WRONG_ONE

NCP Calls

None

NWDPPsmGetFQN

Returns the Fully Qualified Name (FQN) of the NDPS Manager reference
Local Servers: nonblocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_psm.h>

N_EXTERN_LIBRARY (nint) NWDPPsmGetFQN (
   NWDPAccessorRef      accessorRef,
   NWDPPsmRef           psmRef,
   nuint                sizeOfBuffer,
   pNWDPNSrvFQNBuffer   fqnBufferPtr,
   pnuint               sizeOfResultPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

psmRef

(IN) Specifies the reference to the NDPS Manager.

sizeOfBuffer

(IN) Specifies the size of the buffer pointed to by the *labelBufferPtr* parameter.

fqnBufferPtr

(OUT) Points to the resulting FQN (pass the address of the NWDPNSrvFQNBuffer structure).

sizeOfResultPtr

(OUT) Points to the size of the data copied into the *fqnBufferPtr* parameter (optional).

Return Values

0x00000000	N_SUCCESS

0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

Use a FQN to create the NDPS Manager reference or an error is returned.

An alternative method that saves memory is to pass a zero for the *sizeOfBuffer* parameter. The NWDP_LE_RESULT_BUFFER_TOO_SMALL error is returned with the *sizeOfResultPtr* parameter identifying the needed buffer size. Therefore, you need to allocate the amount of memory identified by the *sizeOfResultPtr* parameter, cast the buffer as a NWDPNSrvFQNBuffer structure, and pass it as the *fqnBufferPtr* parameter.

If the **NWDPPsmGetFQN** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x000A0006L NWDP_LE_RESULT_BUFFER_TOO_SMALL
0x000A0008L NWDP_LE_REF_NOT_CREATED_WITH_FQN

NCP Calls

None

See Also

NWDPPsmCreateRefBasedOnFQN

NWDPPsmGetMODObjectAttrSet

Returns the attribute set of the specified object in the MODB (Managed Object Database)

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_psm.h>

N_EXTERN_LIBRARY (nint) NWDPPsmGetMODObjectAttrSet (
   NWDPAccessorRef             accessorRef,
   NWDPPsmRef                  psmRef,
   pNWDPOid                    objectClassPtr,
   pNWDPObjectIdentification   objectIdentPtr,
   NWDPOidSetRef               requestedOidSetRef,
   pNWDPAttrSetRef             resultAttrSetRefPtr,
   pNWDPASAVPRef                resultAvpRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

psmRef

(IN) Specifies the NDPS Manager reference which manages the object.

objectClassPtr

(IN) Points to the object class of the new object which is a NDPS_OC_xxxx definition in the nwdp_wko.ogh file.

objectIdentPtr

(IN) Points to the object identification of the object in the MODB.

requestedOidSetRef

(IN) Specifies the set of attributes that are returned for the object (pass NULL to return all of the attributes for this object).

resultAttrSetRefPtr

(IN) Points to the NWDPAttrSetRef, the resulting attribute set reference of the object.

resultAvpRefPtr

(IN) Points to the NWDPASAVPRef, the resulting attribute and value pointer reference (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

Call the **NWDPASReleaseRef** function to free allocated memory.

If the **NWDPPsmGetMODObjectAttrSet** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN

0x000F0001L NWDP_LE_MODB_NO_SUCH_OBJECT

NCP Calls

None

See Also

NWDPASReleaseRef
NWDPOSCreateRef
NWDPPrtGetMODObjectAttrSet

NWDPPsmListMODObjects

Lists the specified MODB objects and their respective attributes
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_psm.h>

N_EXTERN_LIBRARY (nint) NWDPPsmListMODObjects (
   NWDPAccessorRef               accessorRef,
   NWDPPsmRef                    psmRef,
   pNWDPOid                      objectClassPtr,
   pNWDPASSelector               selectorPtr,
   NWDPOidSetRef                 requestedOidSetRef,
   NWDPASMODObjectListCallback   listCallback,
   nparam                         callerDefinedParam);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

psmRef

(IN) Specifies the reference to the NDPS Manager which manages the objects.

objectClassPtr

(IN) Points to the object class of the new object which is a NDPS_OC_xxxx definition in the nwdp_wko.ogh file.

selectorPtr

(IN) Points to the NWDPASSelector structure used to filter the selection of objects (optional).

requestedOidSetRef

(IN) Specifies the set of attributes returned for each object (pass NULL to return all of the attributes for this object).

listCallBack

(IN) Specifies the application-supplied callback function which is called by the **NWDPPsmListMODObjects** function for each object.

callerDefinedParam

(IN) Specifies a caller-defined parameter, containing application-specific data, that is passed to the callback function (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPPsmListMODObjects** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x030A0001L NWDP_EC_NO_MEMORY
0x000A0005L NWDP_LE_UNKNOWN_CALLBACK_ERR

NCP Calls

None

See Also

NWDPOSCreateRef
NWDPPrtListMODObjects

NWDPPsmListPAs

Lists the printer agents of an NDPS Manager
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_psm.h>

N_EXTERN_LIBRARY (nint) NWDPPsmListPAs (
   NWDPAccessorRef          accessorRef,
   NWDPPsmRef               psmRef,
   NWDPFilterRef            filterRef,
   NWDPPsmPAListCallback    listCallback,
   nparam                   callerDefinedParam);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

psmRef

(IN) Specifies the reference to the NDPS Manager which lists the printer agents.

filterRef

(IN) Specifies the reference to a constructed filter which is used to determine which printer agents are returned (optional).

listCallBack

(IN) Specifies the application-supplied callback function which is called by the **NWDPPsmListPAs** function for each printer agent.

callerDefinedParam

(IN) Specifies a caller-defined parameter, containing application-specific data, that is passed to the callback function (optional).

Return Values

0x0000000	N_SUCCESS

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPPsmListPAs** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x030A0001L NWDP_EC_NO_MEMORY
0x000A0005L NWDP_LE_UNKNOWN_CALLBACK_ERR

NCP Calls

None

See Also

NWDPFltCreateRef
NWDPPsmCreateRefBasedOnAddr
NWDPPsmCreateRefBasedOnFQN

NWDPPsmModifyMODObjectAttrs

Modifies the attributes of an object in the MODB (Managed Object Database)

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_psm.h>

N_EXTERN_LIBRARY (nint) NWDPPsmModifyMODObjectAttrs (
   NWDPAccessorRef             accessorRef,
   NWDPPsmRef                  psmRef,
   pNWDPOid                    objectClassPtr,
   pNWDPObjectIdentification   objectIdentPtr,
   NWDPAttrSetRef              modifyRef,
   NWDPASModifyOperatorEnum     modifyOperator);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

psmRef

(IN) Specifies the reference to the NDPS Manager which manages the object.

objectClassPtr

(IN) Points to the object class of the new object which is a NDPS_OC_xxxx definition in the nwdp_wko.ogh file.

objectIdentPtr

(IN) Points to the object identification of the object that is modified.

modifyRef

(IN) Specifies the reference to the attribute set which is used to modify the object.

modifyOperator

(IN) Specifies the modify operator that is applied to each attribute in the set identified by the *modifyRef* parameter.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

Each attribute in the attribute set specified by the *modifyRef* parameter has a modify operator assigned to it. This modify operator defines the modification which occurs for that attribute. You can set the value for each attribute by using the *qualifierPtr* parameter of the **NWDPASAddAttribute** function. Or, you can overwrite this value by passing a non-NULL (other than NWDP_MODIFY_OP_NULL) value for the *modifyOperator* parameter.

If the **NWDPPsmModifyMODObjectAttrs** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x000F0001L NWDP_LE_MODB_NO_SUCH_OBJECT

NCP Calls

None

See Also

NWDPASAddAttribute
NWDPASCreateRef
NWDPPsmCreateRefBasedOnAddr
NWDPPsmCreateRefBasedOnFQN

NWDPPsmPAListCallback

Lists printer agents
Local Servers: N/A
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_psm.h>

N_TYPEDEF_CALLBACK (nint, NWDPPsmPAListCallback) (
   NWDPAccessorRef      accessorRef,
   nparam               callerDefinedParam,
   nint                 totalCallsToBeMade,
   nint                 currentCallCount,
   pNWDPPsmPAListItem   itemReceivedPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

callerDefinedParam

(IN) Specifies a caller-defined contextual variable.

totalCallsToBeMade

(IN) Specifies the number of times the **NWDPPsmPAListCallback** function is called by the **NWDPPsmListPAs** function (the number of attributes plus one).

currentCallCount

(IN) Specifies the current call number.

itemReceivedPtr

(IN) Points to the NWDPPsmPAListItem structure containing the printer agent information.

Return Values

0x00000000	N_SUCCESS

0xFFFFFFFF	N_FAILURE

Remarks

For the *callerDefinedParam* parameter, this function uses the contextual variable to create the following:

a linked list head into which items can be added

a window handle where items can be displayed

any other data or combination of these two steps needed by the callback routine (optional)

For the *totalCallsToBeMade* parameter, the *itemReceivedPtr* parameter is NULL for the last callback.

NCP Calls

None

See Also

NWDPOidInterpretRef
NWDPPsmListPAs

NWDPPsmRemovePA

Removes a printer agent from the NDPS Manager.
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_psm.h>

N_EXTERN_LIBRARY (nint) NWDPPsmRemovePA (
   NWDPAccessorRef   accessorRef,
   NWDPPsmRef        psmRef,
   pnstr16           paName16Ptr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

psmRef

(IN) Specifies the reference to the NDPS Manager from which the printer agent is removed.

paName16Ptr

(IN) Points to the name of the removed printer agent (this is a Unicode string).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPPsmRemovePA** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN
0x000F0001L NWDP_LE_MODB_NO_SUCH_OBJECT

NCP Calls

None

See Also

NWDPPsmAddPA

NWDPPsmShutdownPA

Shuts down a printer agent.
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_psm.h>

N_EXTERN_LIBRARY (nint) NWDPPsmShutdownPA (
   NWDPAccessorRef       accessorRef,
   NWDPPsmRef            psmRef,
   pnstr16               paName16Ptr,
   pNWDPNameOrOid        reservedPtr,
   NWDPPsmShutdownType   shutdownType);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

psmRef

(IN) Specifies the reference to the NDPS Manager which manages the printer agent that is shutdown.

paName16Ptr

(IN) Points to the name of the shutdown printer agent (this is a Unicode string).

reservedPtr

(IN) Is reserved (pass NULL).

shutdownType

(IN) Specifies the shutdown type.

Return Values

0x00000000	N_SUCCESS
0xFFFFFF	NWDP_RC_INVALID_ACCESSOR

FE	
0xFFFFFFFF	N_FAILURE

Remarks

For the *shutdownType* parameter, pass one of the following:

NWDP_SHUTDOWN_DO_CURRENT_JOBS
NWDP_SHUTDOWN_IMMEDIATE
NWDP_SHUTDOWN_DO_PENDING_JOBS

If the **NWDPPsmShutdownPA** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN

NCP Calls

None

See Also

NWDPPsmStartupPA

NWDPPsmShutdownPSM

Shuts down the NDPS Manager and unloads the NLM.
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_psm.h>

N_EXTERN_LIBRARY (nint) NWDPPsmShutdownPSM (
   NWDPAccessorRef        accessorRef,
   NWDPPsmRef             psmRef,
   pNWDPNameOrOid         reservePtr,
   NWDPPsmShutdownType    shutdownType);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

psmRef

(IN) Specifies the reference to the NDPS Manager that is shutdown.

reservedPtr

(IN) Is reserved (pass NULL).

shutdownType

(IN) Specifies the shutdown type.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

For the *shutdownType* parameter, pass one of the following:

NWDP_SHUTDOWN_DO_CURRENT_JOBS
NWDP_SHUTDOWN_IMMEDIATE
NWDP_SHUTDOWN_DO_PENDING_JOBS

If the **NWDPPsmShutdownPSM** function returns N_FAILURE, NWDPLibErrorMac *(accessorRef)* contains:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN

NCP Calls

None

See Also

NWDPPsmCancelShutdownPSM

NWDPPsmStartupPA

Starts a printer agent
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_psm.h>

N_EXTERN_LIBRARY (nint) NWDPPsmStartupPA (
   NWDPAccessorRef   accessorRef,
   NWDPPsmRef        psmRef,
   pnstr16           paName16Ptr,
   pNWDPNameOrOid    reservedPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

psmRef

(IN) Specifies the reference to the NDPS Manager which manages the printer agent that is started.

paName16Ptr

(IN) Points to the name of the added printer agent (this is a Unicode string).

reservedPtr

(IN) Is reserved (pass NULL).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPPsmStartupPA** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00150001L NWDP_LE_OTHER_NDPS_ERROR_RETURN

NCP Calls

None

See Also

NWDPPsmShutdownPA

NWDPPsmValidateRef

Validates a NDPS Manager reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_psm.h>

N_EXTERN_LIBRARY (nint) NWDPPsmValidateRef (
   NWDPAccessorRef   accessorRef,
   NWDPPsmRef        psmRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

psmRef

(IN) Specifies the NDPS Manager reference to validate.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The **NWDPPsmValidateRef** function verifies that the NDPS Manager reference is associated with the *accessorRef* parameter when the application has received it from an untrusted third party.

If the **NWDPPsmValidateRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER

NCP Calls

None

NWDPResAddResourceFile

Adds a resource file to the Resource Management Service (RMS) data area
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_res.h>

N_EXTERN_LIBRARY (nint) NWDPResAddResourceFile (
   NWDPAccessorRef         accessorRef,
   NWDPResRef              resRef,
   NWDPResourceTypeEnum    resType,
   pnstr                   localDirPtr,
   nparam                  addInput);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

resRef

(IN) Specifies the reference to the resource manager.

resType

(IN) Specifies the resource type.

localDirPtr

(IN) Points to a string containing the directory path of the new resource files or resource index files (for example, A:\).

addInput

(IN) Specifies the type of input to add, based on the *resType* parameter value.

Return Values

0x00000000	N_SUCCESS
0xFFFFFF	NWDP_RC_INVALID_ACCESSOR

FE	
0xFFFFFFFF	N_FAILURE

Remarks

For the *resRef* parameter, the following functions generate a resource reference:

NWDPResCreateRefBasedOnAddr
NWDPResCreateRefBasedOnFQN
NWDPResCreateRefBasedOnSrsSAP

For the *resType* parameter, the resource types include the following:

NWDP_RES_PRN_DRV
NWDP_RES_PRN_DEF
NWDP_RES_BANNER_PAGE
NWDP_RES_FONT

The *addInput* parameter value is based on the following *resType* parameter value:

resType	*addInput*
NWDP_RES_PRN_DRV	Points to the NWDPResAddPrnDrvFile structure
NWDP_RES_PRN_DEF	Points to the NWDPResAddPrnDefFile structure
NWDP_RES_BANNER_PAGE	Points to Unicode string (pnstr16) containing a banner page filename
NWDP_RES_FONT	Points to the NWDPResAddFontFile structure

If the value of the *resType* parameter is NWDP_RES_PRN_DRV or NWDP_RES_PRN_DEF and the filename in the respective *addInput* parameter is an index filename (*.inf or *.ndx), call the **NWDPResAddResourceFile** function to have all files referred to in the index file automatically added to the RMS data area.

When using the **NWDPResAddResourceFile** function to add individual printer driver files to the RMS data area, the driver directory name variable in the NWDPResAddPrnDrvFile structure must contain the "new" Unicode string. The first call to the session creates a new directory. Subsequent calls to the **NWDPResAddResourceFile** function add files to this directory. The session is completed when the driver index file (.inf) is finally added.

If the *accessorRef* parameter returns NWDP_EC_RESMAN_RESULT, the

otherError field in the *accessorRef* parameter contains an error code from the Resource Manager. This error code correlates to the enumeration values found in the nwdp_err.h file.

If the **NWDPResAddResourceFile** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000004L NWDP_EC_STDIO
0x01100001L NWDP_EC_RESMAN_RESULT

NCP Calls

None

See Also

NWDPResCreateRefBasedOnAddr
NWDPResCreateRefBasedOnFQN
NWDPResCreateRefBasedOnSrsSAP

NWDPResCreateRefBasedOnAddr

Creates an Resource Management Service (RMS) reference based on a network address

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_res.h>

N_EXTERN_LIBRARY (nint) NWDPResCreateRefBasedOnAddr (
   NWDPAccessorRef   accessorRef,
   pNWDPNetAddress   addressPtr,
   pNWDPResRef       resRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

addressPtr

(IN) Points to the address of the preferred Resource Manager NLM.

resRefPtr

(OUT) Points to the NWDPResRef containing the resource reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the *accessorRef* parameter returns NWDP_EC_RESMAN_RESULT, the

If the *accessorRef* parameter returns NWDP_EC_RESMAN_RESULT, the *otherError* field in the *accessorRef* parameter contains an error code from the Resource Manager. This error code correlates to the enumeration values found in the nwdp_err.h file.

If the **NWDPResCreateRefBasedOnAddr** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000004L NWDP_EC_STDIO
0x01100001L NWDP_EC_RESMAN_RESULT

NCP Calls

None

See Also

NWDPResCreateRefBasedOnFQN
NWDPResCreateRefBasedOnSrsSAP

NWDPResCreateRefBasedOnFQN

Generates an RMS reference based on its Fully Qualified Name (FQN)
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_res.h>

N_EXTERN_LIBRARY (nint) NWDPResCreateRefBasedOnFQN (
   NWDPAccessorRef   accessorRef,
   pNWDPNSrvFQN      brokerFqnPtr,
   pNWDPResRef       resRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

brokerFqnPtr

(IN) Points to the NWDPNSrvFQN structure containing the FQN of the broker object provided by the name service.

resRefPtr

(OUT) Points to the NWDPResRef, the resource reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the *accessorRef* parameter returns NWDP_EC_RESMAN_RESULT, the

If the *accessorRef* parameter returns NWDP_EC_RESMAN_RESULT, the *otherError* field in the *accessorRef* parameter contains an error code from the Resource Manager. This error code correlates to the enumeration values found in the nwdp_err.h file.

If the **NWDPResCreateRefBasedOnFQN** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000004L NWDP_EC_STDIO
0x01100001L NWDP_EC_RESMAN_RESULT

NCP Calls

None

See Also

NWDPResCreateRefBasedOnAddr
NWDPResCreateRefBasedOnSrsSAP

NWDPResCreateRefBasedOnSrsSAP

Generates an RMS reference by asking the Service Registry Service (SRS) for the nearest Resource Manager address

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_res.h>

N_EXTERN_LIBRARY (nint) NWDPResCreateRefBasedOnSrsSAP (
   NWDPAccessorRef   accessorRef,
   pNWDPResRef       resRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

resRefPtr

(OUT) Points to the NWDPResRef, the resource reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the *accessorRef* parameter returns NWDP_EC_RESMAN_RESULT, the *otherError* field in the *accessorRef* parameter contains an error code from the Resource Manager. This error code correlates to the enumeration values found in the nwdp_err.h file.

If the **NWDPResCreateRefBasedOnSrsSAP** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000004L NWDP_EC_STDIO
0x01100001L NWDP_EC_RESMAN_RESULT

NCP Calls

None

See Also

NWDPResCreateRefBasedOnAddr
NWDPResCreateRefBasedOnFQN

NWDPResDeleteResourceFile

Deletes a resource file in the RMS data area
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_res.h>

N_EXTERN_LIBRARY (nint) NWDPResDeleteResourceFile (
   NWDPAccessorRef         accessorRef,
   NWDPResRef              resRef,
   NWDPResourceTypeEnum    resType,
   nparam                  deleteInput);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

resRef

(IN) Specifies the reference to the resource manager.

resType

(IN) Specifies the resource type.

deleteInput

(IN) Specifies the type of input needed to delete, based on the *resType* parameter value.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

For the *resRef* parameter, the following functions generate a resource reference:

NWDPResCreateRefBasedOnAddr
NWDPResCreateRefBasedOnFQN
NWDPResCreateRefBasedOnSrsSAP

For the *resType* parameter, the resource types are as follows:

NWDP_RES_PRN_DRV
NWDP_RES_PRN_DEF
NWDP_RES_BANNER_PAGE
NWDP_RES_FONT

The current implementation of the **NWDPResDeleteResourceFile** function does not delete all files referenced by an index file (.inf or .ndx).

If the *accessorRef* parameter returns NWDP_EC_RESMAN_RESULT, the *otherError* field in the *accessorRef* parameter contains an error code from the Resource Manager. This error code correlates to the enumeration values found in the nwdp_err.h file.

The *deleteInput* parameter value is based on the following *resType* parameter value:

resType	*deleteInput*
NWDP_RES_PRN_DRV	Points to the NWDPResDeletePrnDrvFile structure
NWDP_RES_PRN_DEF	Points to the NWDPResDeletePrnDefFile structure
NWDP_RES_BANNER_PAGE	Points to Unicode string (pnstr16) containing a banner page filename
NWDP_RES_FONT	Points to the NWDPResDeleteFontFile structure

If the **NWDPResDeleteResourceFile** function returns N_FAILURE, NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000004L NWDP_EC_STDIO
0x01100001L NWDP_EC_RESMAN_RESULT

NCP Calls

None

See Also

NWDPResAddResourceFile
NWDPResCreateRefBasedOnAddr
NWDPResCreateRefBasedOnFQN
NWDPResCreateRefBasedOnSrsSAP

NWDPResDestroyRef

Deletes an RMS reference and frees all associated memory
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_res.h>

N_EXTERN_LIBRARY (nint) NWDPResDestroyRef (
   NWDPAccessorRef   accessorRef,
   pNWDPResRef       resRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

resRefPtr

(IN) Points to the NWDPResRef containing the freed resource reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the *accessorRef* parameter returns NWDP_EC_RESMAN_RESULT, the *otherError* field in the *accessorRef* parameter contains an error code from the Resource Manager. This error code correlates to the enumeration values found in the nwdp_err.h file.

If the **NWDPResDestroyRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000004L NWDP_EC_STDIO
0x01100001L NWDP_EC_RESMAN_RESULT

NCP Calls

None

See Also

NWDPResCreateRefBasedOnAddr
NWDPResCreateRefBasedOnFQN
NWDPResCreateRefBasedOnSrsSAP

NWDPResGetFQN

Obtains the Fully Qualified Name (FQN) from the Resource Management Service (RMS) data area

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_res.h>

N_EXTERN_LIBRARY (nint) NWDPResGetFQN (
   NWDPAccessorRef      accessorRef,
   NWDPResRef           resRef,
   nuint                sizeOfBuffer,
   pNWDPNSrvFQNBuffer   fqnBufferPtr,
   pnuint               sizeOfResultPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

resRef

(IN) Specifies the reference to the resource manager.

sizeOfBuffer

(IN) Specifies the size of the buffer pointed to by the *fqnBufferPtr* parameter.

fqnBufferPtr

(OUT) Points to the resulting FQN.

sizeOfResultPtr

(OUT) Points to the data size copied into the *fqnBufferPtr* parameter (optional).

Return Values

0x00000000	N_SUCCESS

0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

An alternative method which saves memory is to pass a zero for the *sizeOfBuffer* parameter. The NWDP_LE_RESULT_BUFFER_TOO_SMALL error is returned with the *sizeOfResultPtr* parameter identifying the needed buffer size. Therefore, you need to allocate the amount of memory identified by the *sizeOfResultPtr* parameter, cast the buffer as an NWDPNSrvFQNBuffer, and pass it as the *fqnBufferPtr* parameter.

If the **NWDPResGetFQN** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x000A0006L NWDP_LE_RESULT_BUFFER_TOO_SMALL
0x000A0008L NWDP_LE_REF_NOT_CREATED_WITH_FQN

NCP Calls

None

See Also

NWDPResCreateRefBasedOnFQN

NWDPResGetResourceFile

Obtains a file from the RMS data area or from a local disk
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_res.h>

N_EXTERN_LIBRARY (nint) NWDPResGetResourceFile (
   NWDPAccessorRef         accessorRef,
   NWDPResRef              resRef,
   NWDPResourceTypeEnum    resType,
   pnstr                   localSrcDirPtr,
   pnstr                   localDestDirPtr,
   nparam                  getInput,
   time_t                 *fileTimePtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

resRef

(IN) Specifies the reference to the resource manager.

resType

(IN) Specifies the resource type.

localSrcDirPtr

(IN) Points to a string containing the directory path of the new resource files or resource index files (for example, A:\).

localDestDirPtr

(IN) Points to a string containing the directory path of the resource file (for example, C:\WINDOWS\SYSTEM).

getInput

(IN) Specifies the input type based on the value of the *resType* parameter.

fileTimePtr

(OUT) Points to the number of seconds elapsed since 1970 for the

current time stamp of the file (in hexadecimal format).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

For the *resRef* parameter, the following functions generate a resource reference:

NWDPResCreateRefBasedOnAddr
NWDPResCreateRefBasedOnFQN
NWDPResCreateRefBasedOnSrsSAP

For the *resType* parameter, the resource types include the following:

NWDP_RES_PRN_DRV
NWDP_RES_PRN_DEF
NWDP_RES_BANNER_PAGE
NWDP_RES_FONT

If the *accessorRef* parameter returns NWDP_EC_RESMAN_RESULT, the *otherError* field in the *accessorRef* parameter contains an error code from the Resource Manager. This error code correlates to the enumeration values found in the nwdp_err.h file.

The *getInput* parameter value is based on the following *resType* parameter value:

resType	***getInput***
NWDP_RES_PRN_DRV	Points to NWDPResGetPrnDrvFile structure
NWDP_RES_PRN_DEF	Points to NWDPResGetPrnDefFile structure
NWDP_RES_BANNER_PAGE	Points to Unicode string (pnstr16) containing banner page filename
NWDP_RES_FONT	Points to NWDPResGetFontFile structure

If the **NWDPResGetResourceFile** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000004L NWDP_EC_STDIO
0x01100001L NWDP_EC_RESMAN_RESULT

NCP Calls

None

See Also

NWDPResCreateRefBasedOnAddr
NWDPResCreateRefBasedOnFQN
NWDPResCreateRefBasedOnSrsSAP
NWDPResListResource

NWDPResGetResourceFileDate

Obtains the time stamp of a file in the RMS data area
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_res.h>

N_EXTERN_LIBRARY (nint) NWDPResGetResourceFileDate (
   NWDPAccessorRef         accessorRef,
   NWDPResRef              resRef,
   NWDPResourceTypeEnum    resType,
   nparam                  getDateInput,
   time_t                 *fileTimePtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

resRef

(IN) Specifies the reference to the resource manager.

resType

(IN) Specifies the resource type.

getDateInput

(IN) Specifies the input type based on the value of the *resType* parameter.

fileTimePtr

(OUT) Points to the number of seconds elapsed since 1970 for the current time stamp of the file (in hexadecimal format).

Return Values

0x00000000	N_SUCCESS
0xFFFFFF	NWDP_RC_INVALID_ACCESSOR

FE	
0xFFFFFFFF	N_FAILURE

Remarks

For the *resRef* parameter, the following functions generate a resource reference:

NWDPResCreateRefBasedOnAddr
NWDPResCreateRefBasedOnFQN
NWDPResCreateRefBasedOnSrsSAP

For the *resType* parameter, the resource types include the following:

NWDP_RES_PRN_DRV
NWDP_RES_PRN_DEF
NWDP_RES_BANNER_PAGE
NWDP_RES_FONT

If the *accessorRef* parameter returns NWDP_EC_RESMAN_RESULT, the *otherError* field in the *accessorRef* parameter contains an error code from the Resource Manager. This error code correlates to the enumeration values found in the nwdp_err.h file.

The *getDateInput* parameter value is based on the following *resType* parameter value:

resType	*getDateInput*
NWDP_RES_PRN_DRV	Points to the NWDPResGetDatePrnDrvFile structure
NWDP_RES_PRN_DEF	Points to the NWDPResGetDatePrnDefFile structure
NWDP_RES_BANNER_PAGE	Points to Unicode string (pnstr16) containing a banner page filename
NWDP_RES_FONT	Not supported

If the **NWDPResGetResourceFileDate** function returns N_FAILURE, the NWDPLibERrorMac *(accessorRef)* contains the following:

0x01000004L NWDP_EC_STDIO
0x01100001L NWDP_EC_RESMAN_RESULT

NCP Calls

None

See Also

NWDPResCreateRefBasedOnAddr
NWDPResCreateRefBasedOnFQN
NWDPResCreateRefBasedOnSrsSAP
NWDPResListResource

NWDPResListCallback

Returns information through the **NWDPResListResource** function
Local Servers: N/A
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_res.h>

N_TYPEDEF_CALLBACK (nint) NWDPResListCallback (
   NWDPAccessorRef            accessorRef,
   nparam                     listAttributesCallbackParam2,
   nint                       totalCallsToBeMade,
   nint                       currentCallCount,
   pNWDPResListCallbackItem   resListItemPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

listAttributesCallbackParam2

(IN) Specifies a user-defined contextual variable used by the callback routine.

totalCallsToBeMade

(IN) Specifies the number of times to call the **NWDPResListCallback** function.

currentCallCount

(IN) Specifies the number of times the **NWDPResListCallback** function has been called.

resListItemPtr

(IN) Points to the NWDPResListCallbackItem structure.

Return Values

0x00000000	N_SUCCESS

0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

For the *listCallbackParam2* parameter, this function uses the contextual variable to create the following:

a linked list head into which items can be added

a window handle where items can be displayed

any other data or combination of these two steps needed by the callback routine (optional)

If the **NWDPResListCallback** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01100001L NWDP_EC_RESMAN_RESULT

NCP Calls

None

See Also

NWDPResListResource

NWDPResListResource

Obtains a resource list provided by the RMS, a network, or a local disk
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_res.h>

N_EXTERN_LIBRARY (nint) NWDPResListResource (
   NWDPAccessorRef            accessorRef,
   NWDPResRef                 resRef,
   NWDPResourceListTypeEnum   resListType,
   pnstr                      drivePathPtr,
   nparam                     listInput,
   NWDPResListCallback        listCallback,
   nparam                      callerDefinedParam);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

resRef

(IN) Specifies the reference to the resource manager.

resListType

(IN) Specifies the NWDPResListTypeEnum containing an enumeration of the resource type to list.

drivePathPtr

(IN) Points to a string containing the directory path of the new resource files (for example, A:\).

listInput

(IN) Specifies an input type based upon the *resListType* parameter value.

listCallback

(IN) Specifies the **NWDPResListCallback** function receiving the list command results in formats based upon the value of the *resListType* parameter.

callerDefinedParam

(IN) Specifies a user-defined parameter, containing application-specific data, that is passed to the **NWDPResListCallback** function.

Remarks

If the **NWDPResListResource** function is called to list resources on a diskette (such as information on a manufacturer-supplied printer driver diskette), the *drivePathPtr* parameter must point to a string containing the drive path. When calling the **NWDPResListResource** function to list resources in the RMS data storage area, the drivePathPtr must be set to NULL.

For the *resRef* parameter, the following functions generate a resource reference:

NWDPResCreateRefBasedOnAddr
NWDPResCreateRefBasedOnFQN
NWDPResCreateRefBasedOnSrsSAP

The *listInput* parameter value is based on the following *resListType* parameter value:

resListType	***listInput***
NWDP_RES_LIST_PRN_DRV_TYPES	NWDPResOsTypeEnum: NWDP_RES_OS_DOS NWDP_RES_OS_WIN_31 NWDP_RES_OS_WIN_95 NWDP_RES_OS_WIN_NT NWDP_RES_OS_OS_2
NWDP_RES_LIST_PRN_DEF_TYPES	Points to Unicode string (pnstr16) containing the name of a subdirectory in the printer definition section of the Resource Manager storage area. For example, when accessing the Novell printer definition files, the Unicode string "novell" is passed in the *listInput* parameter. This string is used to generate a path into the correct area for the files (for example, \NDPS\RESDIR\PRNDEF\NOVELL).
NWDP_RES_LIST_BANNER_PAGE _FILES	NWDPResBannerTypeEnum: NWDP_RES_BANNER_ALL NWDP_RES_BANNER_PCL NWDP_RES_BANNER_PS NWDP_RES_BANNER_TXT

NWDP_RES_LIST_FONT_TYPES	Points to the NWDPResListFontTypes structure
NWDP_RES_LIST_PRN_DRV_FILES	Points to the NWDPResListDrvFiles structure
NWDP_RES_LIST_PRN_DEF_FILE	Points to the NWDPResListDefFiles structure
NWDP_RES_LIST_FONT_FILES	Points to the NWDPResListFontFiles structure

The return data format is the **NWDPResListCallbackItem** function.

The *callerDefinedParam* parameter contains a list box handle in which data is inserted.

If the *accessorRef* parameter returns NWDP_EC_RESMAN_RESULT, the *otherError* field in the *accessorRef* parameter contains an error code from the Resource Manager. This error code correlates to the enumeration values found in the nwdp_err.h file.

If the **NWDPResListResource** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000004L NWDP_EC_STDIO
0x01100001L NWDP_EC_RESMAN_RESULT

NCP Calls

None

See Also

NWDPResCreateRefBasedOnAddr
NWDPResCreateRefBasedOnFQN
NWDPResCreateRefBasedOnSrsSAP
NWDPResListCallback

NWDPResValidateRef

Validates an RMS reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_res.h>

N_EXTERN_LIBRARY (nint) NWDPResValidateRef (
   NWDPAccessorRef   accessorRef,
   NWDPResRef        resRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

resRef

(IN) Specifies the resource manager reference to validate.

Return Values

0x00000000	N_SUCCESS

Remarks

The **NWDPResValidateRef** function verifies that the resource manager reference is associated with the *accessorRef* parameter when the application has received it from an untrusted third party.

For the *resRef* parameter, the following functions generate a resource reference:

NWDPResCreateRefBasedOnAddr
NWDPResCreateRefBasedOnFQN
NWDPResCreateRefBasedOnSrsSAP

NCP Calls

None

See Also

NWDPResCreateRefBasedOnAddr
NWDPResCreateRefBasedOnFQN
NWDPResCreateRefBasedOnSrsSAP

NWDPSrsCreateRefBasedOnAddr

Creates a service registry reference based on a network address
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_srs.h>

N_EXTERN_LIBRARY (nint) NWDPSrsCreateRefBasedOnAddr (
   NWDPAccessorRef   accessorRef,
   pNWDPNetAddress   netAddrPtr,
   pNWDPSrsRef       srsRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

netAddrPtr

(IN) Points to the NWDPNetAddress structure containing the service registry address.

srsRefPtr

(OUT) Points to the NWDPSrsRef, the resulting service registry reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPSrsCreateRefBasedOnAddr** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPSrsDestroyRef

NWDPSrsCreateRefBasedOnFQN

Creates a service registry reference based on the Fully Qualified Name (FQN)

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_srs.h>

N_EXTERN_LIBRARY (nint) NWDPSrsCreateRefBasedOnFQN (
   NWDPAccessorRef   accessorRef,
   pNWDPNSrvFQN      brokerFqnPtr,
   pNWDPSrsRef       srsRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

brokerFqnPtr

(IN) Points to the NWDPNSrvFQN structure containing the distinguished name of the broker object.

srsRefPtr

(OUT) Points to the NWDPSrsRef, the resulting service registry reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

Remarks

If the **NWDPSrsCreateRefBasedOnFQN** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000005L NWDP_EC_NDS
0x030A0001L NWDP_EC_NO_MEMORY
0x000A000BL NWDP_LE_FQN_NOT_NDPS_BROKER

NCP Calls

None

See Also

NWDPSrsDestroyRef

NWDPSrsCreateRefBasedOnSAP

Creates a service registry reference based on the address of the closest sapping broker

Local Servers: blocking

Remote Servers: blocking

NetWare Server: 4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_srs.h>

N_EXTERN_LIBRARY (nint) NWDPSrsCreateRefBasedOnSAP (
   NWDPAccessorRef   accessorRef,
   pNWDPSrsRef       srsRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

srsRefPtr

(IN) Points to the NWDPSrsRef, the resulting service registry reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPSrsCreateRefBasedOnSAP** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x00050002L NWDP_LE_SRS_NO_SRVC_REG_FOUND

0x030A0001L NWDP_EC_NO_MEMORY

NCP Calls

None

See Also

NWDPSrsDestroyRef

NWDPSrsDeregisterServer

Deregisters a server with the service registry
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_srs.h>

N_EXTERN_LIBRARY (nint) NWDPSrsDeregisterServer (
   NWDPAccessorRef     accessorRef,
   NWDPSrsRef          srsRef,
   NWDPSrsServerType   serverType,
   pNWDPText           serverNamePtr,
   pNWDPNetAddress     addressPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

srsRef

(IN) Specifies the reference to the service registry.

serverType

(IN) Specifies the type of server being deregistered.

serverNamePtr

(IN) Points to the NWDPText structure containing the name of the server being deregistered.

addressPtr

(IN) Points to the address of the server being deregistered.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR

0xFFFFFFFF	N_FAILURE

Remarks

After deregistering a server, that server no longer appears in the callback of the **NWDPSrsListServers** function.

For the *serverType* parameter, pass one of the following:

NWDP_SRS_SERVERTYPE_ALL
NWDP_SRS_SERVERTYPE_PA
NWDP_SRS_SERVERTYPE_NTS
NWDP_SRS_SERVERTYPE_RMS

If the **NWDPSrsDeregisterServer** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x03060002L NWDP_EC_REGISTRY_RESULT

NCP Calls

None

See Also

NWDPSrsRegisterServer

NWDPSrsDestroyRef

Destroys a service registry reference and frees all associated memory
Local Servers: blocking
Remote Servers: blocking
NetWare Server: s4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_srs.h>

N_EXTERN_LIBRARY (nint) NWDPSrsDestroyRef (
   NWDPAccessorRef   accessorRef,
   pNWDPSrsRef       srsRefPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

srsRefPtr

(IN) Points to the NWDPSrsRef, the destroyed service registry reference.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

If the **NWDPSrsDestroyRef** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER

NCP Calls

None

See Also

NWDPSrsCreateRefBasedOnAddr
NWDPSrsCreateRefBasedOnFQN

NWDPSrsGetConnectionStatus

Checks the status of the current connection to the service registry or creates a new connection

Local Servers: blocking

Remote Servers: blocking

NetWare Server: s4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_srs.h>

N_EXTERN_LIBRARY (nint) NWDPSrsGetConnectionStatus (
   NWDPAccessorRef   accessorRef,
   NWDPSrsRef        srsRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

srsRef

(IN) Specifies the reference to the service registry.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

A bind occurs if the *srsRef* parameter is not bound to the service registry of the broker. If the *srsRef* parameter is already bound, the connection is checked. If the connection is lost, NWDP_LE_SRS_CONNECTION_LOST is returned.

If the **NWDPSrsGetConnectionStatus** function returns N_FAILURE, then NWDPLibErrorMac (*accessorRef)* contains the following:

0x00050001L NWDP_LE_SRS_CONNECTION_LOST
0x00050003L NWDP_LE_SRS_NOT_ENABLED
0x03060002L NWDP_EC_REGISTRY_RESULT
0x05180003L NWDP_EC_SRS_BOUND_TO_WRONG_ONE

NCP Calls

None

NWDPSrsGetFQN

Returns the Fully Qualified Name (FQN) of the broker object identified by the service registry reference.

Local Servers: blocking

Remote Servers: blocking

NetWare Server: s4.11

Platform: DOS, NLM, Windows 3.1, Windows 95

SMP Aware: No

Service: Distributed Print

Syntax

```
#include <nwdp_srs.h>

N_EXTERN_LIBRARY (nint) NWDPSrsGetFQN (
   NWDPAccessorRef     accessorRef,
   NWDPSrsRef          srsRef,
   nuint               sizeOfBuffer,
   NWDPNSrvFQNBuffer   fqnBufferPtr,
   pnuint              sizeOfResultPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

srsRef

(IN) Specifies the reference to the service registry of the broker object whose FQN is returned.

sizeOfBuffer

(IN) Specifies the size of the buffer pointed to by the *fqnBufferPtr* parameter.

fqnBufferPtr

(IN/OUT) Points to the resulting FQN (pass the address of a NWDPNSrvFQNBuffer).

sizeOfResultPtr

(OUT) Points to the size of the data copied into the *fqnBufferPtr* parameter (optional).

Return Values

0x0000000	N_SUCCESS

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

An alternative approach that saves memory is to pass a zero for the *sizeOfBuffer* parameter. The NWDP_LE_RESULT_BUFFER_TOO_SMALL error is returned with the *sizeOfResultPtr* parameter identifying the needed buffer size. Therefore, you need to allocate the amount of memory identified by the *sizeOfResultPtr* parameter, cast the buffer as a NWDPNSrvFQNBuffer, and pass it as the *fqnBufferPtr* parameter.

If the **NWDPSrsGetFQN** function returns N_FAILURE, then NWDPLibErrorMac *accessorRef* contains the following:

0x000A0006L NWDP_LE_RESULT_BUFFER_TOO_SMALL
0x000A0008L NWDP_LE_REF_NOT_CREATED_WITH_FQN

NCP Calls

None

See Also

NWDPSrsCreateRefBasedOnFQN

NWDPSrsListServers

Lists the servers known to the service registry
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_srs.h>

N_EXTERN_LIBRARY (nint) NWDPSrsListServers (
   NWDPAccessorRef              accessorRef,
   NWDPSrsRef                   srsRef,
   NWDPSrsServerType            serverType,
   NWDPSrsServersListCallback   listCallback,
   nparam                        callerDefinedParam);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

srsRef

(IN) Specifies the reference to the service registry.

serverType

(IN) Specifies the type of servers that are listed.

listCallback

(IN) Specifies the application-supplied callback function called by the **NWDPSrsListServers** function for each server.

callerDefinedParam

(IN) Specifies a caller-defined parameter, containing application-specific data, that is passed to the callback function (optional).

Return Values

0x00000000	N_SUCCESS

0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

For the *serverType* parameter, pass one of the following:

NWDP_SRS_SERVERTYPE_ALL
NWDP_SRS_SERVERTYPE_PA
NWDP_SRS_SERVERTYPE_NTS
NWDP_SRS_SERVERTYPE_RMS

If the **NWDPSrsListServers** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains:

0x03060002L NWDP_EC_REGISTRY_RESULT
0x000A0005L NWDP_LE_UNKNOWN_CALLBACK_ERR

NCP Calls

None

See Also

NWDPSrsCreateRefBasedOnFQN
NWDPSrsDeregisterServer
NWDPSrsRegisterServer

NWDPSrsRegisterServer

Registers a server with the service registry
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_srs.h>

N_EXTERN_LIBRARY (nint) NWDPSrsRegisterServer (
   NWDPAccessorRef     accessorRef,
   NWDPSrsRef          srsRef,
   NWDPSrsServerType   serverType,
   pNWDPText           serverNamePtr,
   pNWDPNetAddress     addressPtr,
   nuint               infoCount,
   pNWDPInfoItem       infoPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

srsRef

(IN) Specifies the reference to the service registry.

serverType

(IN) Specifies the type of server being registered.

serverNamePtr

(IN) Points to the name of the server that is registered.

addressPtr

(IN) Points to the address of the server that is registered.

infoCount

(IN) Specifies the number of NWDPInfoItems that are pointed to by the *infoPtr* parameter (optional).

infoPtr

(IN) Points to additional information about the server (optional).

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

For the *serverType* parameter, pass one of the following:

NWDP_SRS_SERVERTYPE_ALL
NWDP_SRS_SERVERTYPE_PA
NWDP_SRS_SERVERTYPE_NTS
NWDP_SRS_SERVERTYPE_RMS

The information stored in the *infoPtr* parameter is caller-defined. This information is stored by the broker and is returned during subsequent calls to the **NWDPSrsListServers** function.

If the **NWDPSrsRegisterServer** function returns N_FAILURE, the NWDPLibErrorMac *(accessorRef)* contains the following:

0x03060002L NWDP_EC_REGISTRY_RESULT

NCP Calls

None

See Also

NWDPSrsDeregisterServer

NWDPSrsServersListCallback

Lists the servers
Local Servers: N/A
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_srs.h>

N_TYPEDEF_CALLBACK (nint, NWDPSrsServersListCallback) (
   NWDPAccessorRef       accessorRef,
   nparam                callerDefinedParam,
   nint                  totalCallsToBeMade,
   nint                  currentCallCount,
   pNWDPServerListItem   itemReceivedPtr);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

callerDefinedParam

(IN) Specifies a caller-defined contextual variable.

totalCallsToBeMade

IN) Specifies the number of times the **NWDPBrkListServices** function calls the **NWDPSrsServersListCallback** function (number of servers plus one).

currentCallCount

IN) Specifies the current call number.

itemReceivedPtr

IN) Points to the NWDPServerListItem structure containing the server information.

Return Values

0x00000000	N_SUCCESS

0xFFFFFFFF	N_FAILURE

Remarks

For the *callerDefinedParam* parameter, this function uses the contextual variable to create the following:

a linked list head into which items can be added

a window handle where items can be displayed

any other data or combination of these two steps needed by the callback routine (optional)

For the *totalCallsToBeMade* parameter, the *itemReceivedPtr* parameter is NULL for the last callback.

NCP Calls

None

See Also

NWDPSrsListServers

NWDPSrsValidateRef

Validates a service registry reference
Local Servers: blocking
Remote Servers: blocking
NetWare Server: 4.11
Platform: DOS, NLM, Windows 3.1, Windows 95
SMP Aware: No
Service: Distributed Print

Syntax

```
#include <nwdp_srs.h>

N_EXTERN_LIBRARY (nint) NWDPSrsValidateRef (
   NWDPAccessorRef   accessorRef,
   NWDPSrsRef        srsRef);
```

Parameters

accessorRef

(IN) Points to the NWDPAccessorData structure whose fields are accessed by using the provided error macros.

srsRef

(IN) Specifies the service registry reference to validate.

Return Values

0x00000000	N_SUCCESS
0xFFFFFFFE	NWDP_RC_INVALID_ACCESSOR
0xFFFFFFFF	N_FAILURE

Remarks

The **NWDPSrsValidateRef** function verifies that the service registry reference is associated with the *accessorRef* parameter when the application has received it from an untrusted party.

If the **NWDPSrsValidateRef** function returns N_FAILURE, then NWDPLibErrorMac (*accessorRef*) contains the following:

NWDPLibErrorMac (*accessorRef*) contains the following:

0x01000001L NWDP_EC_INVALID_PARAMETER

NCP Calls

None

Print Service Group

Distributed Print: Structures

NWDPAccessError

Contains access error data

Structure

```
typedef struct {
   NWDPAccessProblem          problem;
   NWDPObjectIdentification   objectIdentification;
   pNWDPNameOrOid             messageOptionPtr;
} NWDPAccessError, N_FAR *pNWDPAccessError;
```

Header File

nwdp_att.h

Fields

problem

Specifies the problem description.

objectIdentification

Specifies the identifier for the object with the problem.

messageOptionPtr

Points to the problem's message explanation. (Optional)

Remarks

Access is determined by rights; the caller must be a trustee and properly authenticated.

NWDPAccessProblem

Contains information about the type of an access problem

Structure

```
typedef struct {
   NWDPProblemTypeEnum   designator;
   union {
      NWDPAccessProblemEnum   standardProblem;
      NWDPObjectIdentifier    extendedProblem;
   } u;
} NWDPAccessProblem, N_FAR *pNWDPAccessProblem;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of problem:

0 NWDP_PROBLEM_TYPE_STANDARD
1 NWDP_PROBLEM_TYPE_EXTENDED

standardProblem

Specifies the type of standard problem:

0 NWDP_ACCESS_WRONG_OBJECT_CLASS
1 NWDP_ACCESS_LACK_ACCESS_RIGHTS
2 NWDP_ACCESS_CANT_INTERRUPT_JOB
3 NWDP_ACCESS_WRONG_OBJECT_STATE
4 NWDP_ACCESS_NOT_BOUND
5 NWDP_ACCESS_NOT_AVAILABLE

extendedProblem

Specifies an OID describing the extended problem.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union.

NWDPAccessorData

Contains public fields of the structure returned by the NWDPLibInit function

Structure

```
typedef struct {
   nuint32                    libError;
   nuint32                    otherError;
   nuint32                    otherError2;
   pnstr8                     otherErrorStrPtr;
   pNWDPErrorReturn           otherNDPSErrorReturnPtr;
   nuint32                    errorLine;
   pnstr8                     errorModule;
   NWDPOidRef                 wellKnownOidRefs;
   pNWDPCommonAccessorData    cADBPtr;
} NWDPAccessorData, N_FAR *pNWDPAccessorData;
```

Header File

nwdp_lib.h

Fields

libError

Specifies the error category code as described in nwdp_err.h. The error category code identifies the library containing the routine.

otherError

Specifies an error code returned from a subordinate library or other numbers whose meaning is clarified by *libError*.

otherError2

Specifies an optional second error code returned from a subordinate library or other numbers whose meaning is clarified by *libError*.

otherErrorStrPtr

Points to an optional string error message returned from a subordinate library or other string data whose meaning is clarified by *libError*.

otherNDPSErrorReturnPtr

Points to an extended error from the NDPSM and is valid only if the *libError* field contains NWDP_LE_OTHER_NDPS_ERROR_RETURN.

errorLine

Specifies the line number within the library module (see *errorModule* below) where the error was reported. This is only valid when using the N_DEBUG version of the library.

errorModule

Specifies the library module where the error was reported. This field plus the *errorLine* (see above) give sufficient information to debug reasons for receiving the error code.This is only valid when using the N_DEBUG version of the library.

wellKnownOidRefs

Contains a pointer to OID references which are permanently allocated within the library. These can be "borrowed" by the application writer by using the macros defined in nwdp_wko.ogh which contain the "r" or "p" prefix in their names ("r" is for an NWDPOidRef and "p" is for a pNWDPOID).

cADBPtr

Contains the "Common Accessor Data Block Pointer." This field refers to several items of data which have only one instance which is shared by multiple accessorRefs. The following fields are made public for the convenience of the library user:

NWDPLocalToUnicodeHandleMac

NWDPUnicodeToLocalHandleMac

NWDPMonocaseHandleMac

Remarks

The condition codes are stored in the structure pointed to by NWDPAccessorRef. The publicly available fields of the NWDPAccessorData structure should be accessed through the following defines:

```
#define NWDPLibErrorMac(accessorRef)
(((NWDPAccessorData *)accessorRef)->libError)
#define NWDPOtherErrorMac(accessorRef)
(((NWDPAccessorData *)accessorRef)->otherError)
#define NWDPOtherError2Mac(accessorRef)
(((NWDPAccessorData *)accessorRef)->otherError2)
#define NWDPOtherErrorStrMac(accessorRef)
(((NWDPAccessorData *)accessorRef)->otherErrorStr)
#define NWDPErrorModuleMac(accessorRef)
(((NWDPAccessorData *)accessorRef)->errorModule)
#define NWDPErrorLineMac(accessorRef)
(((NWDPAccessorData *)accessorRef)->errorLine)
```

The *errorModule* and *errorLine* fields are always available, but might not contain significant data if the debug version of a library subset or DLL is not invoked. These fields are initialized by the **NWDPLibInit** entry point to contain NULL and 0 (zero) respectively. If any other value is found, it is the result of a debug library or DLL attempting to return an error condition.

The *libError* field is always valid and contains either a pertinent library

error code or an "other" error category. Error categories indicate that the error occurred in a call to an underlying library. The library containing the routine is identified by the error category code and the actual error from the library is stored in the *otherError* fields. If the *libError* field does not contain an error category code, the *otherError* field contains no valid data.

The *otherError* field is valid if the NWDP_EC_CATEGORY_MASK (0x01000000) bit is set in the *libError* field. If the NWDP_EC_CATEGORY2_MASK (0x02000000) bit is set in the *libError* field, the *otherError2* field is also valid, indicating the subordinate library also returned a second error value. If the NWDP_EC_CATEGORYSTR_MASK (0x04000000) bit is set along with the NWDP_EC_CATEGORY_MASK (0x01000000) bit, the *otherErrorStrPtr* field points to a null-terminated explanation string returned from the subordinate library.

This string might also be pertinent information rather than an error string. For example if the *libError* field contains NWDP_EC_STDIO and the *otherError2* field contains the platform specific code for a file open failure, then the *otherErrorStrPtr* string might optionally contain the name of the file that could not be opened.

NWDPAddressItem

Contains information about destinations for NDPS event notification

Structure

```
typedef struct {
   NWDPAddressItemTypeEnum   designator;
   union {
      NWDPQualifiedName   userName;
      NWDPQualifiedName   serverName;
      NWDPQualifiedName   volumeName;
      NWDPQualifiedName   orgUnitName;
      NWDPQualifiedName   orgName;
      NWDPQualifiedName   groupName;
      NWDPQualifiedName   dN;
      NWDPQualifiedName   userOrContainerName;
      NWDPText            ceStr;
      NWDPText            ciStr;
      NWDPText            numStr;
      NWDPText            filename;
      NWDPText            phoneNumber;
      nbool               bFlag;
      nint32              integer;
      NWDPNetAddress      netAddress;
      NWDPOctetString     gwUser;
   } u;
} NWDPAddressItem, N_FAR *pNWDPAddressItem;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of item:

0 NWDP_ADDR_ITEM_USER
1 NWDP_ADDR_ITEM_SERVER
2 NWDP_ADDR_ITEM_VOLUME
3 NWDP_ADDR_ITEM_ORG_UNIT
4 NWDP_ADDR_ITEM_ORG
5 NWDP_ADDR_ITEM_GROUP
6 NWDP_ADDR_ITEM_DN
7 NWDP_ADDR_ITEM_USR_OR_CONTAINER
8 NWDP_ADDR_ITEM_CASE_EXACT_STR

9 NWDP_ADDR_ITEM_CASE_IGNORE_STR
10 NWDP_ADDR_ITEM_NUMERIC_STR
11 NWDP_ADDR_ITEM_DOS_FILENAME
12 NWDP_ADDR_ITEM_PHONE_NUMBER
13 NWDP_ADDR_ITEM_BOOLEAN
14 NWDP_ADDR_ITEM_INTEGER
15 NWDP_ADDR_ITEM_NET_ADDRESS
16 NWDP_ADDR_ITEM_ENUM

userName
Specifies the user name.

serverName
Specifies the server name.

volumeName
Specifies the volume name.

orgUnitName
Specifies the organization unit name.

orgName
Specifies the organization name.

groupName
Specifies the group name.

dN
Specifies the NDS distinguished name and tree name.

userOrContainerName
Specifies the user or container name.

ceStr
Specifies a case exact string.

ciStr
Specifies a case ignore string.

numStr
Specifies a numeric (ASCII) string.

filename
Specifies a complete file path.

phoneNumber
Specifies a phone number.

bFlag
Specifies a boolean value.

integer
Specifies an integer value.

netAddress

Specifies a network address.

gwUser

Specifies a GroupWise user ID.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union.

NWDPArea

Defines the size and placement of two-dimensional objects

Structure

```
typedef struct {
   nreal64   minimumX;
   nreal64   maximumX;
   nreal64   minimumY;
   nreal64   maximumY;
} NWDPArea, N_FAR *pNWDPArea;
```

Header File

nwdp_att.h

Fields

minimumX

Specifies a non-negative value for the minimum X coordinate of a printed area.

maximumX

Specifies a non-negative value for the maximum X coordinate of a printed area.

minimumY

Specifies a non-negative value for the minimum Y coordinate of a printed area.

maximumY

Specifies a non-negative value for the maximum Y coordinate of a printed area.

Remarks

The NWDPArea structure contains floating point values which describe the corners of a rectangle.

NWDPAreaSeq

Defines a variable length array of NWDPArea structures

Structure

```
typedef struct {
   nuint       itemCount;
   pNWDPArea   itemPtr;
} NWDPAreaSeq, N_FAR *pNWDPAreaSeq;
```

Header File

nwdp_att.h

Fields

itemCount

Specifies the number of elements in the array.

itemPtr

Points to the first element in the contiguous array.

Remarks

To access each element in order, use the following syntax:

```
nint index;
NWDPAreaSeq array;
for (index = 0; index < array.itemCount; index++)
{
   if (array.itemPtr[index].minimumX == 0)
       break;
}
if (index == array.itemCount)
   printf("Not found.\n");
```

NWDPASAttributeListItem

Contains a set of pointers identifying an attribute and its corresponding value set and OID

Structure

```
typedef struct
{
   pNWDPOid       attributeIdPtr;
   nuint32        qualifier;
   NWDPASAVPRef   ;
} NWDPASAttributeListItem, N_FAR *pNWDPASAttributeListItem;
```

Header File

nwdp_atr.h

Fields

attributeIdPtr

Points to an OID identifying the attribute being returned.

qualifier

Specifies the operation of an attribute modification (add, remove, replace, etc.). The *qualifier* field returns the value of the qualifier found in the attribute itself. This value is set by the list routine which populates the NWDPASAttributeListItem structure.

avpRef

Specifies the attribute in the attribute set and allows nested listing of the attribute values.

Remarks

The NWDPASAttributeListItem structure is used as a parameter for a list callback routine, See the **NWDPASAttributeListCallback** function in the nwdp_atr.h file.

The *qualifier* field is normally unused and should be set to default.

The *avpRef* field returned in the NWDPASAttrListItem structure is the same *avpRef* field passed into the **NWDPASListAttributes** function. Its attribute pointer is updated to point to the current attribute and its *valuePtr* points to the first value of this attribute. Subsequent calls to list the values might change the value pointer.

NWDPASAVPStruct

Contains set of pointers to the "current" attribute and value of an attribute set

Structure

```
typedef struct
{
   NWDPAttrSetRef        attrSetRef;
   pNWDPAttribute        attrPtr;
   pNWDPAttributeValue   valuePtr;
} NWDPASAVPStruct, N_FAR *pNWDPASAVPStruct;
```

Header File

nwdp_atr.h

Fields

attrSetRef

Specifies NWDPAttrSetRef containing a list head.

attrPtr

Points to an attribute within the attribute set.

valuePtr

Points to an attribute value within the attribute pointed to by*attrPtr*.

Remarks

The *attrPtr* and *valuePtr* fields are used to traverse the attribute set. As the set of attributes is traversed using the list calls, the value of*attrPtr* will be positioned to each successive attribute in the set. As the*attrPtr* is changed, the *valuePtr* is also positioned to the first (perhaps only) NWDPAttributeValue in the set belonging to that attribute. This mechanism lends itself to for-loop equivalent techniques using callback functions. If a particular attribute is desired, then use a call to **NWDPASSetAVPByAttributeId**.

Call NWDPASSetAVPByAttributeId to position an AVP to the first value of the corresponding attribute in the set.

NWDPASMODObjectListItem

Contains a set of pointers identifying an object, its class and attributes as maintained by the MOD

Structure

```
typedef struct {
   pNWDPObjectIdentification   objectIdentPtr;
   pNWDPAttributeSet           attrSetPtr;
   pNWDPOid                    objectClassPtr;
} NWDPASMODObjectListItem, N_FAR *pNWDPASMODObjectListItem;
```

Header File

nwdp_atr.h

Fields

objectIdentPtr

Points to the identification for the object.

attrSetPtr

Points to the attribute set containing the object's attributes.

objectClassPtr

Points to the OID representing the object class for the object.

Remarks

The NWDPASMODObjectListItem structure is used as a parameter to a list callback routine. See the **NWDPASMODObjectListCallback** function in the nwdp_atr.h file.

NWDPASSelector

Contains parameters for limiting the results of a ListObjectAttribute (LOA) call to the MOD

Structure

```
typedef struct {
   nuint                          scope;
   NWDPObjectIdentificationSeq    objectIdentSeq;
   NWDPFilterRef                  filterRef;
   nint32                         timeLimit;
   nint32                         countLimit;
} NWDPASSelector, N_FAR *pNWDPASSelector;
```

Header File

nwdp_atr.h

Fields

scope

Specifies the number of levels of contained objects. The value is either 0 (zero, the default) or 1, which indicates that the first level of contained objects should also be returned (job objects would be included when printer objects are requested).

objectIdentSeq

Specifies an array of objects to list attributes from.

filterRef

Specifies a reference to a constructed filter. See **NWDPFltCreateRef**.

timeLimit

Specifies the number of seconds before a continuation response is provided.

countLimit

Specifies the number of objects to include in an individual reply.

Remarks

The NWDPASSelector structure limits the number of items returned in a list of objects/attributes in an individual array.

The *filterRef* field should be NULL if no filtering is required.

NWDPAttribute

Contains an identifier for the attribute and its corresponding value set

Structure

```
typedef struct {
   NWDPObjectIdentifier    attributeId;
   NWDPAttributeValueSet   valueSet;
   nuint32                 qualifier;
} NWDPAttribute, N_FAR *pNWDPAttribute;
```

Header File

nwdp_att.h

Fields

attributeId

Specifies the OID which identifies the attribute.

valueSet

Specifies an array of values; the array might be empty.

qualifier

Specifies a qualifier indicating the type of operation to perform.

Remarks

The qualifier field is set to indicate the type of operation to perform (see **NWDPModifyOperatorEnum** in nwdp_att.h) when calling any of the following:

NWDPASSetModifyOperators
NWDPDocModifyAttrs
NWDPJobModifyAttrs
NWDPNSrvModifyAttrs
NWDPPrtModifyAttrs
NWDPPrtModifyMODObjectAttrs
NWDPPsmModifyMODObjectAttrs

These calls generally allow a blanket change to all attribute structures with a non-zero "operator" parameter.

NWDPAttributeError

Contains information about MOD attribute errors

Structure

```
typedef struct {
   struct {
      nuint                         itemCount;
      pNWDPAttributeProblemItem     itemPtr;
   } problems;
   NWDPObjectIdentification         objectIdentification;
} NWDPAttributeError, N_FAR *pNWDPAttributeError;
```

Header File

nwdp_att.h

Fields

itemCount

Specifies the number of array elements or items.

itemPtr

Points to the first element in the contiguous array.

objectIdentification

Specifies the object to which the attributes belong.

Remarks

To access each element in order, use the following syntax:

```
nint index;
NWDPAttributeItemProblem array;
for (index = 0; index < array.itemCount; index++)
{
   if (array.itemPtr[index].problem == 0)
       break;
}
if (index == array.itemCount)
   printf("Not found.\n");
```

NWDPAttributeProblem

Contains a description of the problem with an attribute

Structure

```
typedef struct {
   NWDPProblemTypeEnum   designator;
   union {
      NWDPAttributeProblemEnum   standardProblem;
      NWDPObjectIdentifier       extendedProblem;
   } u;
} NWDPAttributeProblem, N_FAR *pNWDPAttributeProblem;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of problem:

0 NWDP_PROBLEM_TYPE_STANDARD
1 NWDP_PROBLEM_TYPE_EXTENDED

standardProblem

Specifies the type of standard problem:

0 NWDP_ATTR_INVALID_SYNTAX
1 NWDP_ATTR_UNDEFINED_TYPE
2 NWDP_ATTR_WRONG_MATCHING
3 NWDP_ATTR_CONSTRAINT_VIOLATED
4 NWDP_ATTR_UNSUPPORTED_TYPE
5 NWDP_ATTR_ILLEGAL_MODIFICATION
6 NWDP_ATTR_CONSIST_W_OTHER_ATTR
7 NWDP_ATTR_UNDEFINED_ATTR_VALUE
8 NWDP_ATTR_UNSUPPORTED_VALUE
9 NWDP_ATTR_INVALID_NONCOMPUL_MOD
10 NWDP_ATTR_PER_JOB_INADMISSIBLE
11 NWDP_ATTR_NOT_MULTI_VALUED
12 NWDP_ATTR_MANDATORY_OMITTED
13 NWDP_ATTR_ILLEGAL_FOR_CLASS

extendedProblem

Specifies an OID describing the extended problem.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union.

Attribute problems are caused mostly by syntax issues and should normally be resolved by the developer before the product test cycle is complete.

NWDPAttributeProblemItem

Contains a description of a problem with a specific attribute

Structure

```
typedef struct {
   NWDPAttributeProblem   problem;
   NWDPAttribute          attribute;
   pNWDPNameOrOid         messageOptionPtr;
} NWDPAttributeProblemItem, N_FAR *pNWDPAttributeProblemItem;
```

Header File

nwdp_att.h

Fields

problem

Specifies the problem with the attribute.

attribute

Specifies the OID of the attribute which has the problem.

messageOptionPtr

Points to a message OID or text which describes the problem. (Optional)

Remarks

See NWDPAttributeProblem.

NWDPAttributeSet

Contains a variable length array of NWDPAttribute structures

Structure

```
typedef struct {
   nuint             itemCount;
   pNWDPAttribute    itemPtr;
} NWDPAttributeSet, N_FAR *pNWDPAttributeSet;
```

Header File

nwdp_att.h

Fields

itemCount

Specifies the number of elements in the array.

itemPtr

Points to the first element in the contiguous array.

Remarks

To access each element in order, use the following syntax:

```
nint index;
NWDPAAttribute array;
for (index = 0; index < array.itemCount; index++)
{
   if (array.itemPtr[index].attributeID == 0)
       break;
}
if (index == array.itemCount)
   printf("Not found.\n");
```

NWDPAttributeValue

Allows attribute values to be presented by the print service manager and/or the virtual printer

Structure

```
typedef struct
{
   NWDPAVTenum designator;

   union
   {
      NWDPText                      text;
      NWDPText                      descriptiveName;
      NWDPText                      descriptor;
      NWDPNameOrOid                 message;
      NWDPNameOrOid                 errorMessage;
      NWDPText                      simpleName;
      NWDPDistinguishedNameString   distinguishedNameString;
      NWDPDistinguishedNameStrSeq   distinguishedNameStringSeq;
      nint32                        deltaTime;
      nuint32                       time;
      nint32                        integer;
      NWDPIntegerSeq                integerSeq;
      nuint32                       cardinal;
      NWDPCardinalSeq               cardinalSeq;
      nint32                        positiveInteger;
      NWDPIntegerRange              integerRange;
      NWDPCardinalRange             cardinalRange;
      nint32                        maximumInteger;
      nint32                        minimumInteger;
      nuint64                       cardinal64;
      nreal64                       real;
      NWDPRealSet                   realSeq;
      nreal64                       nonNegativeReal;
      NWDPRealRange                 realRange;
      NWDPRealRange                 nonNegativeRealRange;
      nbool                         boolean;
      nint32                        percent;
      NWDPObjectIdentifier          identifier;
      NWDPObjectIdentifierSet       objectIfdentifierSeq;
      NWDPNameOrOid                 nameOrOid;
      NWDPNameOrOidSet              nameOrOidSeq;
      NWDPText                      distinguishedName;
      NWDPRDNSequence               rdnSequence;
      NWDPRealizationEnum           realization;
      NWDPXYRealDimensions          mediumDimensions;
      pNWDPDimension                dimensionPtr;
      pNWDPXYDimensions             xyDimensionsPtr;
```

```
pNWDPXYDimensions                xyDimensionsPtr;
pNWDPLocations                   locationsPtr;
pNWDPArea                        areaPtr;
NWDPAreaSeq                      areaSeq;
NWDPEdge                         edge;
NWDPText                         fontReference;
NWDPCardinalOrOid                cardinalOrOid;
NWDPOidCardinalMap               oidCardinalMap;
NWDPCardinalOrNameOrOid          caridnalOrNameOrOid;
NWDPPositiveIntegerOrOid         positiveIntegerOrOid;
pNWDPEventHandlingProfile        eventHandlingProfilePtr;
NWDPOctetString                  octetString;
nint32                           jobPriority;
NWDPText                         locale;
pNWDPDeliveryAddrForMethod       deliveryMethodAddressPtr;
pNWDPObjectIdentification        objectIdentificationPtr;
pNWDPResultsProfile              resultsProfilePtr;
NWDPCriteria                     criteria;
NWDPOctetString                  jobPassword;
NWDPJobLevel                     jobLevel;
NWDPJobCategories                jobCategories;
NWDPOctetString                  printCheckpoint;
pNWDPIgnoredAttribute            ignoredAttributePtr;
NWDPResource                     resource;
pNWDPMediumSubstitution          mediumSubstitutionPtr;
NWDPFontSubstitution             fontSubstitution;
NWDPResourceContextSeq           resourceContextSeq;
nint32                           sides;
NWDPPageSelectSequence           pageSelectSeq;
NWDPPageMediaSelect              pageMediaSelect;
NWDPDocumentContent              documentContent;
NWDPPageSize                     pageSize;
NWDPPresentationDirectionEnum    presentationDirection;
NWDPPageOrderTypeEnum            pageOrderType;
NWDPDistinguishedNameString      fileReference;
pNWDPMediumSourceSize            mediumSourceSizePtr;
pNWDPInputTrayMedium             inputTrayMediumPtr;
NWDPOutBinsCharacteristics       outBinsCharacteristics;
NWDPPageIdTypeEnum               pageIdType;
NWDPLevelRange                   levelRange;
pNWDPCategorySet                 categorySetPtr;
NWDPNumbersUpSupported           numbersUpSupported;
pNWDPFinishing                   finishingPtr;
NWDPPrtContainedObjectId         prtContainedObjectId;
pNWDPPrtConfigObjectId           prtConfigObjectIdPtr;
NWDPTypedName                    typedName;
NWDPNetAddress                   netAddress;
NWDPXYDimensionsValue            xyDimensionsValue;
NWDPNameOrOidDimensionMap        nameOrOidDimensionMap;
pNWDPPrinterStateReason          printersStateReasonPtr;
nint32                           enumeration;
NWDPQualifiedName                qualifiedName;
```

```
        NWDPQualifiedName               qualifiedNameSet;
        NWDPColorantSet                 colorantSet;
        pNWDPResourcePrinterId          resourcePrinterIdPtr;
        NWDPEventObjectId               eventObjectId;
        pNWDPQualifiedNameMap           qualifiedNameMapPtr;
        NWDPText                        filePath;
    } u;
} NWDPAttributeValue, N_FAR *pNWDPAttributeValue;
```

Header File

nwdp_att.h

Fields

designator

Specifies the attribute value type:

0 NWDP_AVT_NULL

1 NWDP_AVT_TEXT

2 NWDP_AVT_DESCRIPTIVE_NAME

3 NWDP_AVT_DESCRIPTOR

4 NWDP_AVT_MESSAGE

5 NWDP_AVT_ERROR_MESSAGE

6 NWDP_AVT_SIMPLE_NAME

7 NWDP_AVT_DISTINGUISHED_NAME_STR

8 NWDP_AVT_DIST_NAME_STR_SEQ

9 NWDP_AVT_DELTA_TIME

10 NWDP_AVT_TIME

11 NWDP_AVT_INTEGER

12 NWDP_AVT_INTEGER_SEQ

13 NWDP_AVT_CARDINAL

14 NWDP_AVT_CARDINAL_SEQ

15 NWDP_AVT_POSITIVE_INTEGER

16 NWDP_AVT_INTEGER_RANGE

17 NWDP_AVT_CARDINAL_RANGE

18 NWDP_AVT_MAXIMUM_INTEGER

19 NWDP_AVT_MINIMUM_INTEGER

20 NWDP_AVT_INTEGER_64

21 NWDP_AVT_INTEGER_64_SEQ

22 NWDP_AVT_CARDINAL_64

23 NWDP_AVT_CARDINAL_SEQ

24 NWDP_AVT_POSITIVE_INTEGER_64

25 NWDP_AVT_INTEGER_64_RANGE

26 NWDP_AVT_CARDINAL_64_RANGE

27 NWDP_AVT_MAXIMUM_INTEGER_64

28 NWDP_AVT_MINIMUM_INTEGER_64

29 NWDP_AVT_REAL

30 NWDP_AVT_REAL_SEQ

31 NWDP_AVT_NON_NEGATIVE_REAL

32 NWDP_AVT_REAL_RANGE

33 NWDP_AVT_NON_NEGATIV_REAL_RANGE

34 NWDP_AVT_BOOLEAN

35 NWDP_AVT_PERCENT

36 NWDP_AVT_OBJECT_IDENTIFIER

37 NWDP_AVT_OBJECT_IDENTIFIER_SEQ

38 NWDP_AVT_NAME_OR_OID

39 NWDP_AVT_NAME_OR_OID_SEQ

40 NWDP_AVT_DISTINGUISHED_NAME

41 NWDP_AVT_RDN_SEQUENCE

42 NWDP_AVT_REALIZATION

43 NWDP_AVT_MEDIUM_DIMENSIONS

44 NWDP_AVT_DIMENSION

45 NWDP_AVT_XY_DIMENSIONS

46 NWDP_AVT_LOCATIONS

47 NWDP_AVT_AREA

48 NWDP_AVT_AREA_SEQ

49 NWDP_AVT_EDGE

50 NWDP_AVT_FONT_REFERENCE

51 NWDP_AVT_CARDINAL_OR_OID

52 NWDP_AVT_OID_CARDINAL_MAP

53 NWDP_AVT_CARDINAL_OR_NAM_OR_OID

54 NWDP_AVT_POSITIVE_INT_OR_OID

55 NWDP_AVT_EVENT_HANDLING_PROFILE

56 NWDP_AVT_OCTET_STRING

57 NWDP_AVT_PRIORITY

58 NWDP_AVT_LOCALE

59 NWDP_AVT_METHOD_DELIVERY_ADDR

60 NWDP_AVT_OBJECT_DIENTIFICATION

61 NWDP_AVT_RESULTS_PROFILE

62 NWDP_AVT_CRITERIA

63 NWDP_AVT_JOB_PASSWORD

64 NWDP_AVT_JOB_LEVEL

65 NWDP_AVT_JOB_CATEGORIES

66 NWDP_AVT_PRINT_CHECKPOINT

67 NWDP_AVT_IGNORED_ATTRIBUTE

68 NWDP_AVT_RESOURCE

69 NWDP_AVT_MEDIUM_SUBSTITUTION

70 NWDP_AVT_FONT_SUBTITUTION

71 NWDP_AVT_RESOURCE_CONTEXT_SEQ

72 NWDP_AVT_SIDES

73 NWDP_AVT_PAGE_SELECT_SEQ

74 NWDP_AVT_PAGE_MEDIA_SELECT

75 NWDP_AVT_DOCUMENT_CONTENT

76 NWDP_AVT_PAGE_SIZE

77 NWDP_AVT_PRESENTATION_DIRECTION

78 NWDP_AVT_PAGE_ORDER

79 NWDP_AVT_FILE_REFERENCE

80 NWDP_AVT_MEDIUM_SOURCE_SIZE

81 NWDP_AVT_INPUT_TRAY_MEDIUM

82 NWDP_AVT_OUTPUT_BINS_CHARS

83 NWDP_AVT_PAGE_ID_TYPE

84 NWDP_AVT_LEVEL_RANGE

85 NWDP_AVT_CATEGORY_SET

86 NWDP_AVT_NUMBERS_UP_SUPPORTED

87 NWDP_AVT_FINISHING

88 NWDP_AVT_PRT_CONTAINED_OBJ_ID

89 NWDP_AVT_PRT_CONFIG_OBJECT_ID

90 NWDP_AVT_TYPED_NAME

91 NWDP_AVT_NET_ADDRESS

92 NWDP_AVT_XY_DIMENSIONS_VALUE

93 NWDP_AVT_NAME_OR_OID_DIM_MAP

94 NWDP_AVT_PRINTER_STATE_REASON

95 NWDP_AVT_ENUMERATION

96 NWDP_AVT_QUALIFIED_NAME

97 NWDP_AVT_QUALIFIED_NAME_SET

98 NWDP_AVT_COLORANT_SET

99 NWDP_AVT_RESOURCE_PRINTER_ID

100 NWDP_AVT_EVENT_OBJECT_ID

101 NWDP_AVT_QUALIFIED_NAME_MAP

101 NWDP_AVT_FILE_PATH

text

(NWDP_AVT_TEXT) Specifies text rendered only as level 2 Unicode (see ISO10175-1: section 9.1.5.1).

descriptiveName

(NWDP_AVT_DESCRIPTIVE_NAME) Specifies a descriptive name (see ISO10175-1: section 9.1.5.2).

descriptor

(NWDP_AVT_DESCRIPTOR) Specifies a descriptor intended for the locale of the descriptor's creator (see ISO10175-1: section 9.1.5.3).

message

(NWDP_AVT_MESSAGE) Specifies a human readable message generated by a user for another user (see ISO10175-1: section 9.1.5.4).

errorMessage

(NWDP_AVT_ERROR_MESSAGE) Specifies a human readable message generated by the server for users (see ISO10175-1: section 9.1.5.5).

simpleName

(NWDP_AVT_SIMPLE_NAME) Specifies a short human readable string representation of a simple name for an object created by an administrator (see ISO10175-1: section 9.1.5.6).

distinguishedNameString

(NWDP_AVT_DISTINGUISHED_NAME_STR) Specifies the name of an object, file, or person as a string that can be either a simple name, or a simple name qualified with a path name (see ISO10175-1: section 9.1.5.7).

distinguishedNameStringSeq

(NWDP_AVT_DIS_NAME_STR_SEQ) Specifies a sequence of distinguished name string values (see ISO10175-1: section 9.1.5.8).

deltaTime

(NWDP_AVT_DELTA_TIME) Specifies an integer value for a period of time measured in seconds (see ISO10175-1: section 9.1.5.9).

time

(NWDP_AVT_TIME) Specifies a value for a calendar date and time of day (see ISO10175-1: section 9.1.5.10).

integer

(NWDP_AVT_INTEGER) Specifies an integer value (see ISO10175-1: section 9.1.5.11).

integerSeq

(NWDP_AVT_INTEGER_SEQ) Specifies a sequence of integer values (see ISO10175-1: section 9.1.5.12).

cardinal

(NWDP_AVT_CARDINAL) Specifies a non-negative integer value

(see ISO10175-1: section 9.1.5.13).

cardinalSeq

(NWDP_AVT_CARDINAL_SEQ) Specifies a sequence of non-negative integer values (see ISO10175-1: section 9.1.5.14).

positiveInteger

(NWDP_AVT_POSITIVE_INTEGER) Specifies a non-zero positive integer value (see ISO10175-1: section 9.1.5.15).

integerRange

(NWDP_AVT_INTEGER_RANGE) Specifies a range of integer values (see ISO10175-1: section 9.1.5.16).

cardinalRange

(NWDP_AVT_CARDINAL_RANGE) Specifies a range of non-negative integer values (see ISO10175-1: section 9.1.5.17).

maximumInteger

(NWDP_AVT_MAXIMUM_INTEGER) Specifies the maximum allowable value for an integer (see ISO10175-1: section 9.1.5.18).

minimumInteger

(NWDP_AVT_MINIMUM_INTEGER) Specifies the minimum allowable value for an integer (see ISO10175-1: section 9.1.5.19).

cardinal64

(NWDP_AVT_CARDINAL_64) Specifies a non-negative 64-bit integer value (see ISO10175-1: section 9.1.5.22).

real

(NWDP_AVT_REAL) Specifies a non-integer value (see ISO10175-1: section 9.1.5.29).

realSeq

(NWDP_AVT_REAL_SEQ) Specifies a sequence of non-integer values (see ISO10175-1: section 9.1.5.30).

nonNegativeReal

(NWDP_AVT_NON_NEGATIVE_REAL) Specifies a non-negative, non-integer value (see ISO10175-1: section 9.1.5.31).

realRange

(NWDP_AVT_REAL_SEQ) Specifies a range of allowable values for a Real datatype (see ISO10175-1: section 9.1.5.32).

nonNegativeRealRange

(NWDP_AVT_NON_NEGATIV_REAL_RANGE) Specifies a range of allowable values for a non-negative, non-integer datatype (see ISO10175-1: section 9.1.5.33).

boolean

(NWDP_AVT_BOOLEAN) Specifies a Boolean value (see ISO10175-1: section 9.1.5.34).

percent

(NWDP_AVT_PERCENT) Specifies a percentage value (see ISO10175-1: section 9.1.5.35).

identifier

(NWDP_AVT_IDENTIFIER) Specifies an object identifier (see ISO10175-1: section 9.1.5.36).

objectIdentifierSeq

(NWDP_AVT_OBJECT_IDENTIFIER_SEQ) Specifies a sequence of object identifiers (see ISO10175-1: section 9.1.5.37).

nameOrOid

(NWDP_AVT_NAME_OR_OID) Specifies an identifier that might have either a global or local form (see ISO10175-1: section 9.1.5.38).

nameOrOidSeq

(NWDP_AVT_NAME_OR_OID_SEQ) Specifies a sequence of *nameOrOid* values (see ISO10175-1: section 9.1.5.39).

distinguishedName

(NWDP_AVT_DISTINGUISHED_NAME) Specifies a distinguished name (see ISO10175-1: section 9.1.5.40).

rdnSequence

(NWDP_AVT_RDN_SEQUENCE) Specifies a relative distinguished name (see ISO10175-1: section 9.1.5.41).

realization

(NWDP_AVT_REALIZATION) Specifies whether an object is classified as a logical or physical object or both (see ISO10175-1: section 9.1.5.42):

0 NWDP_REALIZATION_LOGICAL

1 NWDP_REALIZATION_PHYSICAL

2 NWDP_REALIZATION_LOG_AND_PHYS

mediumDimensions

(NWDP_AVT_MEDIUM_DIMENSIONS) Specifies the size of a medium in millimeters (see ISO10175-1: section 9.1.5.43).

dimensionPtr

(NWDP_AVT_DIMENSION) Specifies a linear dimension either in millimeters or as a name or OID with an optional tolerance in millimeters (see ISO 10175-1: section 9.1.5.44).

xyDimensionsPtr

(NWDP_AVT_DIMENSIONS) Specifies a rectangular area either in millimeters or as a name or OID with an optional tolerance in millimeters (see ISO10175-1: section 9.1.5.45).

locationsPtr

(NWDP_AVT_LOCATIONS) Specifies a sequence of locations as distances from a reference edge in millimeters with an optional tolerance in millimeters (see ISO10175-1: section 9.1.5.46).

areaPtr

(NWDP_AVT_AREA) Specifies the size and placement of two dimensional objects with respect to the Reference Coordinate System (RCS) (see ISO10175-1: section 9.1.5.47).

areaSeq

(NWDP_AVT_AREA_SEQ) Specifies a sequence of area values (see ISO10175-1: section 9.1.5.48).

edge

(NWDP_AVT_EDGE) Specifies the edge of a medium with respect to the Reference Coordinate System (RCS) (see ISO10175-1: section 9.1.5.49):

0 NWDP_EDGE_BOTTOM

1 NWDP_EDGE_RIGHT

2 NWDP_EDGE_TOP

3 NWDP_EDGE_LEFT

fontReference

(NWDP_AVT_FONT_REFERENCE) Specifies a type font (see ISO10175-1: section 9.1.5.50).

cardinalOrOid

(NWDP_AVT_CARDINAL_OR_OID) Specifies a value as a cardinal or an OID that normally names a cardinal (see ISO10175-1: section 9.1.5.51).

oidCardinalMap

(NWDP_AVT_OID_CARDINAL_MAP) Specifies a value as a cardinal and also have an OID that acts as an alias for the cardinal (see ISO10175-1: section 9.1.5.52).

cardinalOrNameOrOid

(NWDP_AVT_CARDINAL_OR_NAM_OR_OID) Specifies a value either as a cardinal or a NameOrOid that normally names a cardinal (see ISO10175-1: section 9.1.5.53).

positiveIntegerOrOid

(NWDP_AVT_POSITIVE_INT_OR_OID) Specifies either an object identifier or a positive integer (see ISO10175-1: section 9.1.5.54).

eventHandlingProfilePtr

(NWDP_AVT_EVENT_HANDLING_PROFILE) Specifies an event handling profile (see ISO10175-1: section 9.1.5.55).

octetString

(NWDP_AVT_OCTET_STRING) Specifies an octet string.

jobPriority

(NWDP_AVT_PRIORITY) Specifies a priority for a print job (see ISO10175-1: section 8.1.2.1).

locale

(NWDP_AVT_LOCALE) Specifies the location used by the server when processing text (see ISO10175-1: section 8.1.2.3).

deliveryMethodAddressPtr

(NWDP_AVT_METHOD_DELIVERY_ADDR) Specifies the default delivery addresses for various delivery methods that might be required for notification or logging (see ISO10175-1: section 8.1.2.4).

objectIdentificationPtr

(NWDP_AVT_OBJECT_IDENTIFICATION) Specifies the type of object identifier (see ISO10175-1: section 8.2.4.1):

0 NWDP_OBJ_ID_NONE
1 NWDP_OBJ_ID_PRT_CONTAIND_OBJ_ID
2 NWDP_OBJ_ID_DOCUMENT_IDENTIFIER
3 NWDP_OBJ_ID_OBJECT_IDENTIFIER
4 NWDP_OBJ_ID_OBJECT_NAME
5 NWDP_OBJ_ID_NAME_OR_OID
6 NWDP_OBJ_ID_SIMPLE_NAME
7 NWDP_OBJ_ID_PRT_CONFIG_OBJ_ID
8 NWDP_OBJ_ID_QUALIFIED_NAME
9 NWDP_OBJ_ID_EVENT_OBJECT_ID

resultsProfilePtr

(NWDP_AVT_RESULTS_PROFILE) Specifies the job sets that comprise a job's output (see ISO10175-1: section 9.2.2.1).

criteria

(NWDP_AVT_CRITERIA) Specifies a list of limit values that control job processing (see ISO10175-1: section 9.2.3.1 to 9.2.3.3).

jobPassword

(NWDP_AVT_JOB_PASSWORD) Specifies a password should be entered before a print job starts (see ISO10175-1: section 9.2.4.4).

jobLevel

(NWDP_AVT_JOB_LEVEL) Specifies a confidentiality level and the organization associated with the level (see ISO10175-1: section 9.2.7.1).

jobCategories

(NWDP_AVT_JOB_CATEGORIES) Specifies a job's set of confidentiality categories and the organization associated with the categories (see ISO10175-1: section 9.2.7.2).

printCheckpoint

(NWDP_AVT_PRINT_CHECKPOINT) Specifies the job copies, document copies, pages, and octets completed for the document or documents on the specified printers and the local context information

at which the last checkpoint was taken (see ISO10175-1: section 9.2.8.17).

ignoredAttributePtr

(NWDP_AVT_IGNORED_ATTRIBUTE) Specifies each non-compulsory attribute type or value ignored (see ISO10175-1: section 9.2.8.29).

resource

(NWDP_AVT_RESOURCE) Specifies resources other than fonts that the server can use as the resource default for the pages of the document requiring a specification (see ISO10175-1: section 9.3.2.4).

mediumSubstitutionPtr

(NWDP_AVT_MEDIUM_SUBSTITUTION) Specifies a list of pairs of medium identifiers; each pair identifies a medium followed by its substitute (see ISO10175-1: section 9.3.2.9).

fontSubstitution

(NWDP_AVT_FONT_SUBSTITUTION) Specifies a list of pairs of font references; each pair identifies a font followed by its substitute (see ISO10175-1: section 9.3.2.11).

resourceContext

(NWDP_AVT_RESOURCE_CONTEXT) Specifies a sequence of one or more contexts for the document (see ISO10175-1: section 9.3.2.12).

sides

(NWDP_AVT_SIDES) Specifies the number of printable surfaces of the medium being printed. Possible values are 1 and 2. (See ISO10175-1: section 9.3.2.19).

pageSelectSeq

(NWDP_AVT_PAGE_SELECT_SEQ) Specifies one or more sequences of pages to be printed, specified by ordinal page number, alphanumeric string, and/or a tag of a group of pages (see ISO10175-1: section 9.3.2.20):

0 NWDP_PAGE_ID_NOMINAL_NUMBER

1 NWDP_PAGE_ID_ALPHANUMERIC

2 NWDP_PAGE_ID_TAG

pageMediaSelect

(NWDP_AVT_PAGE_MEDIA_SELECT) Specifies which pages should be printed on the specified media (see ISO10175-1: section 9.3.2.21).

documentContentPtr

(NWDP_AVT_DOCUMENT_CONTENT) Specifies a transfer-method-specific reference for the document to be transferred (see ISO10175-1: section 9.3.3.4):

0 NWDP_DOC_CONTENT_INCLUDED

1 NWDP_DOC_CONTENT_REFERENCED

pageSizePtr

(NWDP_AVT_PAGE_SIZE) Specifies the page size for which pages in the document have been formatted (see ISO10175-1: section 9.3.4.1).

presentationDirection

(NWDP_AVT_PRESENTATION_DIRECTION) Specifies the most significant presentation direction of text contained in the document (see ISO10175-1: section 9.3.4.3):

0 NWDP_DIR_TO_RIGHT_TO_BOTTOM

1 NWDP_DIR_TO_LEFT_TO_BOTTOM

2 NWDP_DIR_BIDIRECT_TO_BOTTOM

3 NWDP_DIR_TO_RIGHT_TO_TOP

4 NWDP_DIR_TO_LEFT_TO_TOP

5 NWDP_DIR_BIDIRECT_TO_TOP

6 NWDP_DIR_TO_BOTTOM_TO_RIGHT

7 NWDP_DIR_TO_BOTTOM_TO_LEFT]

8 NWDP_DIR_TO_TOP_TO_LEFT

9 NWDP_DIR_TO_TOP_TO_RIGHT

pageOrderType

(NWDP_AVT_PAGE_ORDER) Specifies the page order for which pages in the document have been formatted (see ISO10175-1: section 9.3.4.4):

0 NWDP_PAGE_ORDER_UNKNOWN

1 NWDP_PAGE_ORDER_FIRST_TO_LAST

2 NWDP_PAGE_ORDER_LAST_TO_FIRST

fileReference

(NWDP_AVT_FILE_REFERENCE) Specifies a file reference or a transfer-method-specific reference (see ISO10175-1: section 9.3.5.4)

mediumSourceSizePtr

(NWDP_AVT_MEDIUM_SOURCE_SIZE) Specifies the sizes of media that are supported by the printer (see ISO10175-1: section 9.4.16).

inputTrayMediumPtr

(NWDP_AVT_INPUT_TRAY_MEDIUM) Specifies the medium in each input tray of the printer (see ISO10175-1: section 9.4.34).

outBinsCharacterIstics

(NWDP_AVT_OUTPUT_BINS_CHARS) Specifies the characteristics for each output bin for the printer (see ISO10175-1: section 9.4.37):

0 NWDP_PAGE_FACE_UNKNOWN
1 NWDP_PAGE_FACE_UP
2 NWDP_PAGE_FACE_DOWN

pageIdType

(NWDP_AVT_PAGE_ID_TYPE) Specifies the type of page identifier supported by this printer (see ISO10175-1: section 9.4.51):

0 NWDP_PAGE_ID_TYPE_NUMERIC

1 NWDP_PAGE_ID_TYPE_ALPHANUMERIC

2 NWDP_PAGE_ID_TYPE_TAG

levelRange

(NWDP_AVT_LEVEL_RANGE) Specifies the printer's minimum and maximum confidentiality levels. Each confidentiality level range is associated with an organization which defines the meaning of the level (see ISO10175-1: section 9.4.61.1).

categorySetPtr

(NWDP_AVT_CATEGORY_SET) Specifies the printer's minimum and maximum sets of confidentiality categories. Each confidentiality category set is associated with an organization which defines the meaning of the categories (see ISO10175-1: section 9.4.61.2).

numbersUpSupported

(NWDP_AVT_NUMBERS_UP_SUPPORTED) Specifies the number up values and imposition objects supported by the printer (see ISO10175-1: section 9.4.62.2):

0 NWDP_NUMBERS_UP_CARDINAL
1 NWDP_NUMBERS_UP_NAME_OR_OID
2 NWDP_NUMBERS_UP_CARDINAL_RANGE

finishingPtr

(NWDP_AVT_FINISHING) Specifies the type of type of finishing objects supported by this printer (see ISO10175-1: section 9.4.27 and 9.13.8):

0 NWDP_FINISHING_PROC_STITCHING
1 NWDP_FINISHING_PROC_BINDING
2 NWDP_FINISHING_PROC_TRIMMING
3 NWDP_FINISHING_PROC_DIE_CUTTING
4 NWDP_FINISHING_PROC_PUNCHING
5 NWDP_FINISHING_PROC_PERFORATING
6 NWDP_FINISHING_PROC_SLITTING
7 NWDP_FINISHING_PROC_INSERT
8 NWDP_FINISHING_PROC_COVER
9 NWDP_FINISHING_PROC_FOLDING
10 NWDP_FINISHING_PROC_OTHER

prtContainedObjectId

(NWDP_AVT_CONTAINED_OBJ_ID) Specifies an object contained by a logical printer.

prtConfigObjectIdPtr

(NWDP_AVT_CONFIG_OBJECT_ID) Specifies a printer configuration object that describes defaults and limits.

typedName

(NWDP_AVT_TYPED_NAME) Specifies a structure whose fields correspond one-to-one with the NDS Typed_Name_T structure, which contains an NDS object name, and two integer fields named *Level* and *Interval*.

netAddress

(NWDP_AVT_NET_ADDRESS) Specifies an address structure whose fields correspond to those of the NDSNet_Address_T structure whose *addressType* field of type NET_ADDRESS_TYPE corresponds one-to-one with the following:

0 NWDP_NET_IPX
1 NWDP_NET_IP
2 NWDP_NET_SDLC
3 NWDP_NET_TOKENRING_ETHERNET
4 NWDP_NET_OSI
5 NWDP_NET_APPLETALK
6 NWDP_NET_COUNT

xyDimensionsValuePtr

(NWDP_AVT_XY_DIMENSIONS_VALUE) Points to a structure that specifies both the type of the value and the value itself:

0 NWDP_DIM_XY_REAL
1 NWDP_DIM_XY_NAMED
2 NWDP_DIM_XY_CARDINAL

nameOrOidDimensionMapPtr

(NWDP_AVT_NAME_OR_OID_DIM_MAP) Points to an array of mappings between pairs of dimensions and Names or OIDs, with one dimension pair per name.

printerStateReasonPtr

(NWDP_AVT_PRINTER_STATE_REASON) Points to the reason for the state of the printer. For example, the state might be "warning" while the reason for the warning is "toner low." This reason is identified by the Name or OID used as the "identifier" field of the structure.

enumeration

(NWDP_AVT_ENUMERATION) Specifies that the integer field of the enumeration has a textual interpretation. This interpretation is acquired by creating a reference to the OID which identifies the

attribute and then calling **NWDPOidInterpretRefValue**, passing in the enumeration value as well as the NWDPOidRef. The resulting string is the explanation of the enumeration value.

qualifiedName

(NWDP_AVT_QUALIFIED_NAME) Specifies an object's name, either as a simple name (Unicode zero-terminated string) or as a fully canonicalized NDS name with corresponding NDS Tree name.

qualifiedNameSet

(NWDP_AVT_QUALIFIED_NAME_SET) Specifies a set of qualified names.

colorantSetPtr

(NWDP_AVT_COLORANT_SET) Specifies the type of colorant set (Xerox color extensions):

0 NWDP_COLORANT_SET_NAME
1 NWDP_COLORANT_SET_DESCR

resourcePrinterId

(NWDP_AVT_RESOURCE_PRINTER_ID) Specifies the strings sufficient to locate within the NDPS Resource Manager (supported within the NDPS Broker) the Printer's resources, i.e. driver files.

eventObjectIdPtr

(NWDP_AVT_EVENT_OBJECT_ID) Points to the identification of an NDPS event. This includes the event type, the name of the event, and the OID of the Containing Class.

qualifiedNameMapPtr

(NWDP_AVT_QUALIFIED_NAME_MAP) Points to a pairing of qualified names.

filePath

(NWDP_AVT_FILE_PATH) Specifies a Unicode string (null terminated) which gives a complete path to a file.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union.

NWDPAttributeValueSet

Contains a variable length array of NWDPAttributeValue structures

Structure

```
typedef struct {
   nuint                  itemCount;
   pNWDPAttributeValue    itemPtr;
} NWDPAttributeValueSet, N_FAR *pNWDPAttributeValueSet;
```

Header File

nwdp_att.h

Fields

itemCount

Specifies the number of elements in the array.

itemPtr

Points to the first element in the contiguous array.

Remarks

To access each element in order, use the following syntax:

```
nint index;
NWDPAttributeValue array;
for (index = 0; index < array.itemCount; index++)
{
   if (array.itemPtr[index].text == 0)
       break;
}
if (index == array.itemCount)
   printf("Not found.\n");
```

NWDPAttrSetStruct

Contains an attribute set

Structure

```
typedef struct {
   NWDPAttributeSet   attrSet;
   nuint              reserved1;
   nptr               reserved2;
   nuint              reserved3;
} NWDPAttrSetStruct, N_FAR *pNWDPAttrSetStruct;
```

Header File

nwdp_atr.h

Fields

attrSet

Specifies an attribute set.

reserved1

This field is reserved for internal use.

reserved2

This field is reserved for internal use.

reserved3

This field is reserved for internal use.

Remarks

Exposes the NWDPAttributeSet contained within. This structure is pointed to by NWDPAttrSetRef. NWDPAttrSetStruct is the structure pointed to by the NWDPAVPRef. Only the *attrSet* field is considered "public" information; all other fields within the structure are names reserved because they are manipulated only by calls such as **NWDPASSetAVPByAttributeId**.

NWDPBindingSpec

Contains a binding specification

Structure

```
typedef struct {
   NWDPBindingEnum         designator;
   union {
      NWDPNameOrOid        namedBinding;
      NWDPBindParameters   parameters;
   } u;
} NWDPBindingSpec, N_FAR *pNWDPBindingSpec;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of binding:

0 NWDP_BINDING_NAMED

1 NWDP_BINDING_PARAMETERS

namedBinding

Specifies a named binding that corresponds to ISO 10175-1: section 9.13.8.6.1.

parameters

Specifies a set of binding parameters including reference size, reference edge, jog edge, binding type binding type and binding color. Corresponds to ISO 10175-1: sections 9.13.8.6.2 to 9.13.8.6.4.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPBindingSpec corresponds to ISO 10175-1: section 9.13.8.6

NWDPBindParameters

Contains a description of binding parameters

Structure

```
typedef struct {
   NWDPCommonParameters   common;
   NWDPNameOrOid          bindType;
   NWDPNameOrOid          bindColor;
} NWDPBindParameters, N_FAR *pNWDPBindParameters;
```

Header File

nwdp_att.h

Fields

common

Corresponds to ISO 10175-1: section 9.13.8.6.2.

bindType

Corresponds to ISO 10175-1: section 9.13.8.6.3.

bindColor

Corresponds to ISO 10175-1: section 9.13.8.6.4.

Remarks

NWDPBindParameters corresponds to ISO 10175-1: section 9.13.8.6.2.

NWDPBrokerService

Defines whether a service is enabled or disabled

Structure

```
typedef struct {
   NWDPServiceTypeEnum   serviceType;
   nbool                 enabledSwitch;
} NWDPBrokerService, N_FAR *pNWDPBrokerService;
```

Header File

nwdp_bkr.h

Fields

serviceType

Specifies the type of service:

0 NWDP_SERVICE_TYPE_SRS (Service Registry Service)

1 NWDP_SERVICE_TYPE_ENS (Event Notification Service)

2 NWDP_SERVICE_TYPE_RMS (Resource Management Service)

enabledSwitch

N_TRUE implies the service is enabled.

Remarks

None

NWDPCardinalOrNameOrOid

Contains a cardinal or NameOrOid value

Structure

```
typedef struct {
   NWDPCardinalOrNameOrOidEnum designator;
   union {
      nint32          cardinal;
      NWDPNameOrOid   nameOrOid;
   } u;
} NWDPCardinalOrNameOrOid, N_FAR *pNWDPCardinalOrNameOrOid;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type:

0 NWDP_CARDINAL_OR_NAME_OR_OID_NUM
1 NWDP_CARDINAL_OR_NAME_OR_OID_ID

cardinal

Specifies a cardinal value.

nameOrOid

Specifies a nameOrOid value.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPCardinalOrNameOrOid corresponds to ISO10175-1: section 9.1.5.53.

NWDPCardinalOrOid

Contains a cardinal or OID value

Structure

```
typedef struct {
   NWDPCardinalOrOidEnum     designator;
   union {
      nint32                 cardinal;
      NWDPObjectIdentifier   oid;
   } u;
} NWDPCardinalOrOid, N_FAR *pNWDPCardinalOrOid;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type:

0 NWDP_CARDINAL_OR_OID_NUMBER
1 NWDP_CARDINAL_OR_OID_ID

cardinal

Specifies a cardinal value.

oid

Specifies an OID.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPCardinalOrOid corresponds to ISO10175-1: section 9.1.5.51.

NWDPCardinalRange

Contains a range of non-negative integer values

Structure

```
typedef struct {
   nuint32   lowerBound;
   nuint32   upperBound;
} NWDPCardinalRange, N_FAR *pNWDPCardinalRange;
```

Header File

nwdp_att.h

Fields

lowerBound

Specifies the lower boundary of the value range.

upperBound

Specifies the upper boundary of the value range.

Remarks

NWDPCardinalRange corresponds to ISO 10175-1: section 9.1.5.17.

NWDPCardinal64Range

Contains a range of 64-bit non-negative values

Structure

```
typedef struct {
   nuint64   lowerBound;
   nuint64   upperBound;
} NWDPCardinal64Range, N_FAR *pNWDPCardinal64Range;
```

Header File

nwdp_att.h

Fields

lowerBound

Specifies the lower boundary of the value range.

upperBound

Specifies the upper boundary of the value range.

Remarks

NWDPCardinal64Range corresponds to ISO 10175-1: section 9.1.5.26.

NWDPCardinalSeq

Contains a variable length array of cardinal values

Structure

```
typedef struct {
   nuint      itemCount;
   pnuint32   itemPtr;
} NWDPCardinalSeq, N_FAR *pNWDPCardinalSeq;
```

Header File

nwdp_att.h

Fields

itemCount

Specifies the number of elements in the array.

itemPtr

Points to the first element in the contiguous array.

Remarks

NWDPCardinalSeq corresponds with ISO 10175-1: section 9.1.5.14.

NWDPCategorySet

Contains sets of confidentiality categories

Structure

```
typedef struct {
   NWDPObjectIdentifier   organizationId;
   NWDPCardinalSeq        categoryMinimum;
   NWDPCardinalSeq        categoryMaximum;
} NWDPCategorySet, N_FAR *pNWDPCategorySet;
```

Header File

nwdp_att.h

Fields

organizationId

Specifies the organization identifier.

categoryMinimum

Specifies the minimum set of confidentiality categories.

categoryMaximum

Specifies the maximum set of confidentiality categories.

Remarks

NWDPCategorySet corresponds to ISO10175-1: section 9.4.61.2.

NWDPColorantSet

Contains information about a colorant set

Structure

```
typedef struct {
   NWDPColorantEnum          designator;
   union {
      NWDPNameOrOid          colorantSetName;
      NWDPColorantSetDescr   colorantSetDescr;
   } u;
} NWDPColorantSet, N_FAR *pNWDPColorantSet;
```

Header File

nwdp_att.h

Fields

designator

Specifies the colorant set type:

0 NWDP_COLORANT_SET_NAME
1 NWDP_COLORANT_SET_DESCR

colorantSetName

Specifies the name of the colorant set.

colorantSetDescr

Specifies a description of the colorant set.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPColorantSet describes the Xerox color extensions of ISO10175-1.

NWDPColorantSetDescr

Contains a description of a colorant set

Structure

```
typedef struct {
   NWDPObjectIdentifier   colorantSetClass;
   NWDPNameOrOidSet       colorantSpecSeq;
} NWDPColorantSetDescr, N_FAR *pNWDPColorantSetDescr;
```

Header File

nwdp_att.h

Fields

colorantSetClass

Specifies the colorant class.

colorantSpecSeq

Specifies the colorant.

Remarks

NWDPColorantSet describes the Xerox color extensions of ISO10175-1.

NWDPCommonAccessorData

Contains localization handles

Structure

```
typedef struct {
   nptr   localToUnicodeHandle;
   nptr   unicodeToLocalHandle;
   nptr   monocaseHandle;
} NWDPCommonAccessorData, N_FAR *pNWDPCommonAccessorData;
```

Header File

nwdp_lib.h

Fields

localToUnicodeHandle

Specifies the handle returned by the **NWGetLocaleToUnicodeHandle** function.

unicodeToLocalHandle

Specifies the handle returned by the **NWGetUnicodeToLocaleHandle** function.

monocaseHandle

Specifies the handle returned by the **NWGetMonocaseHandle** function.

Remarks

The library must have these available for its own use during the entire time the accessor is valid. Macros are provided to access the fields within this structure so the library's clients can omit calling the **NWGetUnicodeToLocalHandle** etc. entry points.

See Also

NWDPAccessorData

NWDPCommonParameters

Contains generic finishing process parameters

Structure

```
typedef struct {
   pNWDPXYDimensions   referenceSizeOptionPtr;
   NWDPEdgeOption      referenceEdgeOption;
   NWDPEdgeOption      jogEdgeOption;
} NWDPCommonParameters, N_FAR *pNWDPCommonParameters;
```

Header File

nwdp_att.h

Fields

referenceSizeOptionPtr

Points to the reference size, which specifies the nominal medium size in millimeters for all calculations in the associated finishing process.

referenceEdgeOption

Specifies the reference edge, which is the edge of the document relative to which a finishing process is applied.

jogEdgeOption

Specifies the jog edge, which is a second edge of a document along which instances of the media are aligned.

Remarks

NWDPCommonParameters corresponds to ISO 10175-1: section 9.13.1.

NWDPCoverParameters

Contains finishing process cover parameters

Structure

```
typedef struct {
   NWDPCommonParameters   common;
   pNWDPNameOrOid         frontCoverOptionPtr;
   pNWDPNameOrOid         backCoverOptionPtr;
} NWDPCoverParameters, N_FAR *pNWDPCoverParameters;
```

Header File

nwdp_att.h

Fields

common

Specifies generic finishing process parameters; corresponds to ISO 10175-1: section 9.13.8.13.2.

frontCoverOptionPtr

Specifies a front cover; corresponds to ISO 10175-1: section 9.13.8.13.3.

backCoverOptionPtr

Specifies a back cover; corresponds to ISO 10175-1: section 9.13.8.13.4.

Remarks

NWDPCoverParameters corresponds to ISO 10175-1: section 9.13.8.13.2 to 9.13.8.13.4.

NWDPCoverSpec

Contains a finishing process cover specification

Structure

```
typedef struct {
   NWDPCovertSpecEnum       designator;
   union {
      NWDPNameOrOid         namedCovers;
      NWDPCoverParameters   parameters;
   } u;
} NWDPCoverSpec, N_FAR *pNWDPCoverSpec;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of cover specification:

0 NWDP_COVER_SPEC_NAMED
1 NWDP_COVER_SPEC_PARAMETERS

namedCovers

Specifies a named covers identifier; corresponds to ISO 10175-1: section 9.13.8.13.1.

parameters

Specifies a set of cover parameters; corresponds to ISO 10175-1: section 9.13.8.13.2.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union.

NWDPCoverSpec corresponds to ISO 10175-1: section 9.13.8.13.

NWDPCredentials

Contains information about user access privileges

Structure

```
typedef struct {
   NWDPCredentialsEnum   designator;
   union {
      NWDPCreds        simple;
      struct {
         nuint         itemCount;
         pnint8        itemPtr;
      } certified;
      NWDPNdpsCred0    ndpsCred0;
   } u;
} NWDPCredentials, N_FAR *pNWDPCredentials;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of credentials:

0 NWDP_CREDENTIALS_SIMPLE
1 NWDP_CREDENTIALS_CERTIFIED
2 NWDP_CREDENTIALS_NDPS_0

simple

Specifies a simple credential to be used when there is no result from calling the NDS **NWDSWhoAmI** function.

certified.itemCount

Specifies the length of the item.

certified.itemPtr

Points to the item.

ndpsCred0

Specifies yourself indirectly by telling the receiving agent a server to which you are connected and the connection number.

Remarks

There is a one-to-one correspondence between the designator values and

the fields within the union. NWDPCredentials corresponds to ISO 10175-1: section 7.1.1.

NWDPCreds

Contains information about user access privileges

Structure

```
typedef struct {
   NWDPText    name;
   struct {
      nuint    itemCount;
      pnint8   itemPtr;
   } password;
} NWDPCreds, N_FAR *pNWDPCreds;
```

Header File

nwdp_att.h

Fields

name

Specifies the user name.

password.itemCount

Specifies the size of the password.

password.itemPtr

Points to the password.

Remarks

NWDPCreds corresponds to ISO 10175-1: section 7.1.1; it is not used to current NDPS implementations because it is a "simple" credential and is only used to identify an empty name and password.

NWDPCriteria

Contains criteria for aborting a print job

Structure

```
typedef struct {
   NWDPObjectIdentifier     type;
   NWDPCriterionThreshold   threshold;
} NWDPCriteria, N_FAR *pNWDPCriteria;
```

Header File

nwdp_att.h

Fields

type

Specifies a count, or a time-related attribute associated with the processing of a print job.

threshold

Specifies a value used by the server to determine when to abort a print job.

Remarks

NWDPCriteria corresponds to ISO 10175-1: section 9.2.3.1.

NWDPCriterionThreshold

Contains information used by the server to determine when to abort a print job

Structure

```
typedef struct {
   NWDPAVTEnum   designator;
   union {
      nint32    deltaTime;
      nuint32   time;
      nint32    integer;
      nuint32   cardinal;
      nint32    positiveInteger;
      nint32    percent;
   } u;
} NWDPCriterionThreshold, N_FAR *pNWDPCriterionThreshold;
```

Header File

nwdp_att.h

Fields

designator

Specifies the attribute value type:

9 NWDP_AVT_DELTA_TIME
10 NWDP_AVT_TIME
11 NWDP_AVT_INTEGER
13 NWDP_AVT_CARDINAL
15 NWDP_AVT_POSITIVE_INTEGER
35 NWDP_AVT_PERCENT

deltaTime

Specifies an integer value for a period of time measured in seconds (see ISO10175-1: section 9.1.5.9).

time

Specifies a value for a calendar date and time of day (see ISO10175-1: section 9.1.5.10).

integer

Specifies an integer value (see ISO10175-1: section 9.1.5.11).

cardinal

Specifies a non-negative integer value (see ISO10175-1: section 9.1.5.13).

positiveInteger

Specifies a non-zero positive integer value (see ISO10175-1: section 9.1.5.15).

percent

Specifies a percentage value (see ISO10175-1: section 9.1.5.35).

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPCriterionThreshhold corresponds to ISO 10175-1: section 9.2.3.1.

NWDPDeliveryAddress

Contains information that identifies the recipient of event notification

Structure

```
typedef struct {
   NWDPDeliveryAddressEnum   designator;
   union {
      NWDPText                      mhsAddress;
      NWDPText                      distinguishedName;
      NWDPText                      text;
      NWDPOctetString               octetString;
      NWDPDistinguishedNameString   distinguishedNameString;
      NWDPQualifiedName             qualifiedName;
   } u;
} NWDPDeliveryAddress, N_FAR *pNWDPDeliveryAddress;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of delivery address:

0 NWDP_DA_OR_ADDR_AND_DIR_NAME
1 NWDP_DA_DISTINGUISHED_NAME
2 NWDP_DA_TEXT
3 NWDP_DA_OCTET_STRING
4 NWDP_DA_DIST_NAME_STRING
5 NWDP_DA_RPC_ADDRESS
6 NWDP_DA_QUALIFIED_NAME`

mhsAddress

Specifies an address in MHS format.

distinguishedName

Specifies a distinguished name.

text

Specifies a text string.

octetString

Specifies an octet string.

distinguishedNameString

Specifies a distinguished name.

qualifiedName

Specifies a qualified name, which is the NDS fully canonicalized name and tree name combination.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPDeliveryAddress corresponds to ISO 10175-1: section 9.1.5.55.

NWDPDeliveryAddrForMethod

Associates a delivery method with an address in a list of default pairings

Structure

```
typedef struct {
   NWDPObjectIdentifier   method;
   NWDPDeliveryAddress    address;
} NWDPDeliveryAddrForMethod, N_FAR *pNWDPDeliveryAddrForMethod;
```

Header File

nwdp_att.h

Fields

method

Specifies the delivery method.

address

Specifies the default delivery address.

Remarks

NWDPDeliveryAddrForMethod corresponds to ISO 10175-1: section 8.1.2.4.

NWDPDeliveryAddrItem

Contains an event notification delivery address

Structure

```
typedef struct {
   NWDPAddressItemTypeEnum   designator;
   union {
      NWDPRestrictionInfo   userRestrict;
      NWDPRestrictionInfo   serverRestrict;
      NWDPRestrictionInfo   volumeRestrict;
      NWDPRestrictionInfo   orgUnitRestrict;
      NWDPRestrictionInfo   orgRestrict;
      NWDPRestrictionInfo   groupRestrict;
      NWDPRestrictionInfo   dNRestrict;
      NWDPRestrictionInfo   userOrContainerRestrict;
      NWDPRestrictionInfo   caseExactStrRestrict;
      NWDPRestrictionInfo   caseIgnoreStrRestrict;
      NWDPRestrictionInfo   numericStrRestrict;
      NWDPRestrictionInfo   dosFileNameRestrict;
      NWDPRestrictionInfo   phoneNumberRestrict;
      NWDPRestrictionInfo   booleanRestrict;
      NWDPRestrictionInfo   integerRestrict;
      NWDPRestrictionInfo   netAddressRestrict;
      struct {
                     nuint   itemCount;
                 pNWDPText   itemPtr;
      ) choiceSet;
      NWDPRestrictionInfo   gwUserRestrict;
   } u;
} NWDPDeliveryAddrItem, N_FAR *pNWDPDeliveryAddrItem;
```

Header File

nwdp_nto.h

Fields

designator

Specifies the type of item:

0 NWDP_ADDR_ITEM_USER
1 NWDP_ADDR_ITEM_SERVER
2 NWDP_ADDR_ITEM_VOLUME
3 NWDP_ADDR_ITEM_ORG_UNIT
4 NWDP_ADDR_ITEM_ORG
5 NWDP_ADDR_ITEM_GROUP

6 NWDP_ADDR_ITEM_DN
7 NWDP_ADDR_ITEM_USR_OR_CONTAINER
8 NWDP_ADDR_ITEM_CASE_EXACT_STR
9 NWDP_ADDR_ITEM_CASE_IGNORE_STR
10 NWDP_ADDR_ITEM_NUMERIC_STR
11 NWDP_ADDR_ITEM_DOS_FILENAME
12 NWDP_ADDR_ITEM_PHONE_NUMBER
13 NWDP_ADDR_ITEM_BOOLEAN
14 NWDP_ADDR_ITEM_INTEGER
15 NWDP_ADDR_ITEM_NET_ADDRESS
16 NWDP_ADDR_ITEM_ENUM

userRestrict

Specifies an NDS user object name restriction.

serverRestrict

Specifies an NDS server object name restriction.

volumeRestrict

Specifies an NDS volume object name restriction.

orgUnitRestrict

Specifies an NDS organizational unit, for example, "OU=MySuborganization".

orgRestrict

Specifies an NDS organization restriction, for example, "O=CompanyName".

groupRestrict

Specifies an NDS Group restriction.

dNRestrict

Specifies an NDS distinguished name restriction.

userOrContainerRestrict

Specifies an NDS user object or NDS container, for example, Organization or Organizational Unit.

caseExactStrRestrict

Specifies a string whose matching rules require exact matches of upper/lower case in spelling.

caseIgnoreStrRestrict

Specifies a string whose matching rules require case-insensitive spelling.

numericStrRestrict

Specifies a numeric string restriction.

dosFileNameRestrict

Specifies a DOS file name restriction.

phoneNumberRestrict

Specifies a phone number restriction.

booleanRestrict

Specifies a Boolean value restriction.

integerRestrict

Specifies an integer value restriction.

netAddressRestrict

Specifies a network address restriction.

itemCount

Specifies the number of elements in the array.

itemPtr

Points to the first item in the contiguous array.

gwUserRestrict

Specifies a GroupWise user restriction.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union.

NWDPDieCuttingParameters

Contains definitions of die cutting parameters

Structure

```
typedef struct {
   NWDPCommonParameters   common;
   NWDPXYDimensions       dieCutLocation;
   NWDPNameOrOid          dieCutName;
} NWDPDieCuttingParameters, N_FAR *pNWDPDieCuttingParameters;
```

Header File

nwdp_att.h

Fields

common

Specifies generic die cutting parameters; corresponds to ISO 10175-1: section 9.13.8.8.2.

dieCutLocation

Specifies a die cut location as an xyDimensions type; corresponds to ISO 10175-1: section 9.13.8.8.3.

dieCutName

Specifies the name of a die cut to apply to the die cut finishing process; corresponds to ISO 10175-1: section 9.13.8.8.4.

Remarks

NWDPDieCuttingParameters corresponds to ISO 10175-1: section 9.13.8.8.2.

NWDPDieCuttingSpec

Specifies die cutting parameters by name or by parameters

Structure

```
typedef struct {
   NWDPDieCuttingEnum   designator;
   union {
      NWDPNameOrOid              namedDieCutting;
      NWDPDieCuttingParameters   parameters;
   } u;
} NWDPDieCuttingSpec, N_FAR *pNWDPDieCuttingSpec;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of die cutting parameters:

0 NWDP_DIE_CUTTING_NAMED
1 NWDP_DIE_CUTTING_PARAMETERS

namedDieCutting

Specifies the name of a die cutting operation; corresponds to ISO 10175-1: section 9.13.8.8.1.

parameters

Specifies a die cutting parameter set; corresponds to ISO 10175-1: section 9.13.8.8.2.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPDieCuttingSpec corresponds to ISO 10175-1: section 9.13.8.8.

NWDPDimension

Specifies a single linear dimension in millimeters or as a name or OID with an optional tolerance in millimeters

Structure

```
typedef struct {
   NWDPDimValue     value;
   NWDPRealOption   toleranceOption;
} NWDPDimension, N_FAR *pNWDPDimension;
```

Header File

nwdp_att.h

Fields

value

Specifies a linear dimension in millimeters.

toleranceOption

Specifies a dimension tolerance in millimeters. (Optional)

Remarks

NWDPDimension corresponds to ISO 10175-1: section 9.1.5.44.

NWDPDimValue

Contains part of the NWDPDimension structure implemented as an independent type

Structure

```
typedef struct {
   NWDPDimValueEnum   designator;
   union {
      nreal64         numericValue;
      NWDPNameOrOid   namedValue;
   } u;
} NWDPDimValue, N_FAR *pNWDPDimValue;
```

Header File

nwdp_att.h

Fields

designator

Specifies an enumeration containing the dimension type:

0 NWDP_DIMENSION_NUMERIC
1 NWDP_DIMENSION_NAMED

numericValue

Specifies a numeric value.

namedValue

Specifies a named value. For example, the name "Dozen" might be mapped to the numeric value 12 to allow "Dozen" to be understood as a dimension value.

Remarks

There is a one-to-one correspondence between the designator values and the fields in the union.

NWDPDistinguishedNameString

Allows distinguished names to be passed

Structure

```
typedef struct
{
   NWDPText          name;
   pNWDPNameOrOid    syntaxOptionPtr;
} NWDPDistinguishedNameString, N_FAR *pNWDPDistinguishedNameString;
```

Header File

nwdp_att.h

Fields

name

Specifies a name formatted in any of the possible forms of NWDPText.

syntaxOptionPtr

Points to NWDPNameOrOid containing an optional syntax identifier (or NULL for NDS default). For example, ID_VAL_DN_SYNTAX_X_500 meaning X.500 name syntax.

Remarks

NWDPDistinguishedNameString corresponds to ISO 10175-1: section 9.1.5.7. However, with the advent of support for multiple trees, NWDPDistinguishedNameString is not sufficient; use NWDPQualifiedName instead.

NWDPDistinguishedNameStrSeq

Specifies a variable length array of NWDPDistinguishedNameString structures

Structure

```
typedef struct {
   nuint                          itemCount;
   pNWDPDistinguishedNameString   itemPtr;
} NWDPDistinguishedNameStrSeq, N_FAR *pNWDPDistinguishedNameStrSeq;
```

Header File

nwdp_att.h

Fields

itemCount

Specifies the number of elements in the array.

itemPtr

Points to the first element in the contiguous array.

Remarks

NWDPDistinguishedNameStrSeq corresponds to ISO 10175-1: section 9.1.5.8.

To access each element in order, use the following syntax:

```
nint index;
NWDPDistinguishedNameString array;
for (index = 0; index < array.itemCount; index++)
{
   if (array.itemPtr[index].name == 0)
       break;
}
if (index == array.itemCount)
   printf("Not found.\n");
```

NWDPDocAccessError

Describes a problem with document access

Structure

```
typedef struct {
   NWDPDocAccessProblem       problem;
   NWDPObjectIdentification   objectIdentification;
   pNWDPNameOrOid             messageOptionPtr;
} NWDPDocAccessError, N_FAR *pNWDPDocAccessError;
```

Header File

nwdp_att.h

Fields

problem

Specifies the type of problem.

objectIdentification

Specifies the document with the problem.

messageOptionPtr

Points to an optional explanation of the problem.

Remarks

NWDPDocAccessError corresponds to ISO 10175-1: section 8.4.3.

NWDPDocAccessProblem

Describes a problem with document access

Structure

```
typedef struct {
   NWDPProblemTypeEnum   designator;
   union {
      NWDPDocAccessProblemEnum   standardProblem;
      NWDPObjectIdentifier       extendedProblem;
   } u;
} NWDPDocAccessProblem, N_FAR *pNWDPDocAccessProblem;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of problem:

0 NWDP_PROBLEM_TYPE_STANDARD
1 NWDP_PROBLEM_TYPE_EXTENDED

standardProblem

Specifies the type of standard problem:

0 NWDP_DOC_ACCESS_NOT_AVAILABLE
1 NWDP_DOC_ACCESS_FIDELI_TIME_EXP
2 NWDP_DOC_ACCESS_DENIED
3 NWDP_DOC_ACCESS_UNKNOWN_DOC
4 NWDP_DOC_ACCESS_NO _DOCS_IN_JOB

extendedProblem

Specifies an OID describing the extended problem.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPDocAccessProblem corresponds to ISO 10175-1: section 8.4.3.

NWDPDocumentContent

Contains a transfer-method-specific reference for a document being transferred

Structure

```
typedef struct {
   NWDPDocumentContentEnum   designator;
   union {
      NWDPOctetString               includedDocument;
      NWDPDistinguishedNameString   referencedDocument;
   } u;
} NWDPDocumentContent, N_FAR *pNWDPDocumentContent;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of document content:

0 NWDP_DOC_CONTENT_INCLUDED
1 NWDP_DOC_CONTENT_REFERENCED

includedDocument

Specifies an included document. An included document is one that is actually part of this structure, represented by a length of bytes (*itemCount* field) which contains the printer-ready data and a pointer (*itemPtr* field) to that data.

referencedDocument

Specifies a referenced document.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPDocumentContent corresponds to ISO 10175-1: section 9.3.3.4.

NWDPDocumentIdentifier

Identifies a document as part of a specific job

Structure

```
typedef struct {
   NWDPPrtContainedObjectId   jobIdentifier;
   nuint32                    documentNumber;
} NWDPDocumentIdentifier, N_FAR *pNWDPDocumentIdentifier;
```

Header File

nwdp_att.h

Fields

jobIdentifier

Specifies a print job.

documentNumber

Specifies the document within the print job. Number one is the first document of the job.

Remarks

NWDPDocumentIdentifier corresponds to ISO 10175-1: section 9.2.1.1.

NWDPEdgeOption

Contains part of a finishing process specification

Structure

```
typedef struct {
   nint32     flag;
   NWDPEdge   value;
} NWDPEdgeOption, N_FAR *pNWDPEdgeOption;
```

Header File

nwdp_att.h

Fields

flag

Specifies a Boolean value indicating whether the *value* field is valid.

value

Specifies the type of edge:

0 NWDP_EDGE_BOTTOM
1 NWDP_EDGE_RIGHT
2 NWDP_EDGE_TOP
3 NWDP_EDGE_LEFT

Remarks

NWDPEdgeOption corresponds to ISO 10175-1: sections 9.13.8.3.4 and 9.13.8.3.5.

A non-zero *flag* field indicates the optional edge (either "reference" or "jog") is present. A "reference" edge defaults to left if the flag field is zero. A `jog" edge defaults to bottom if the "reference" edge is left or right and left if the "reference" edge is top or bottom.

NWDPErrorReturn

Describes abstract errors associated with abstract operations

Structure

```
typedef struct {
   NWDPErrorTypeEnum   designator;
   union {
      NWDPSecurityError    securityError;
      struct {
         nuint               itemCount;
         pNWDPServiceError   itemPtr;
      } serviceErrorSeq;
      struct {
         nuint              itemCount;
         pNWDPAccessError   itemPtr;
      } accessErrorSeq;
      NWDPPrinterError     printerError;
      struct {
         nuint                 itemCount;
         pNWDPSelectionError   itemPtr;
      } selectionErrorSeq;
      NWDPDocAccessError   docAccessError;
      NWDPAttributeError   attributeError;
      NWDPUpdateError      updateError;
   } u;
} NWDPErrorReturn, N_FAR *pNWDPErrorReturn;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of error:

0 NWDP_ERROR_TYPE_SECURITY
1 NWDP_ERROR_TYPE_SERVICE
2 NWDP_ERROR_TYPE_ACCESS
3 NWDP_ERROR_TYPE_PRINTER
4 NWDP_ERROR_TYPE_SELECTION
5 NWDP_ERROR_TYPE_DOCUMENT_ACCESS
6 NWDP_ERROR_TYPE_ATTRIBUTE
7 NWDP_ERROR_TYPE_UPDATE

securityError

Specifies a security error.

itemCount

Specifies the number of array elements or items.

itemPtr

Points to the first element in the contiguous array.

itemCount

Specifies the number of array elements or items.

itemPtr

Points to the first element in the contiguous array.

printerError

Specifies a printer error.

itemCount

Specifies the number of array elements or items.

itemPtr

Points to the first element in the contiguous array.

docAccessError

Specifies a document access error.

attributeError

Specifies an attribute error.

updateError

Specifies an update error.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPErrorReturn corresponds to ISO 10175-1: section 8.4.

NWDPEventHandlingProfile

Defines how notification of a specified event should be handled

Structure

```
typedef struct {
   nuint32                  profileId;
   NWDPPersistenceEnum      persistence;
   NWDPQualifiedName        consumerName;
   NWDPOctetString          supplierId;
   nint32                   languageId;
   NWDPNameOrOid            methodId;
   NWDPNotifyDeliveryAddr   deliveryAddress;
   NWDPEventObjectSet       eventObjectSet;
   NWDPQualifiedName        accountOption;
} NWDPEventHandlingProfile, N_FAR *pNWDPEventHandlingProfile;
```

Header File

nwdp_att.h

Fields

profileId

Specifies the temporary ID of the registered profile that generated the event report. This ID is useful in removing and/or modifying the profile contained at the Notification Server.

persistence

Specifies the type of persistence:

0 NWDP_PERSISTENCE_PERMANENT
1 NWDP_PERSISTENCE_VOLATILE

consumerName

Specifies the owner of the profile.

supplierId

Specifies an application-defined parameter, in which the caller can place information to store in the profile and return when the profile is retrieved.

languageId

Specifies the language to use in rendering the message. If NWDP_LID_DEFAULT is used, notifications will use the language identified by the environment variable NWLANGUAGE.

methodId

Specifies the method used by the notification service.

Specifies the method used by the notification service.

deliveryAddress

Specifies the locations where notifications will be sent. Use **NWDPNtfyListPrompts** to determine how to fill in the NWDPAddressItem structure for a given method.

eventObjectSet

Specifies the set of objects which present the list of events for which there is interest.

accountOption

This field is reserved for future use. Set the designator to NWDP_QUALIFIED_NAME_NONE.

Remarks

NWDPEventHandlingProfile is used to manage requests for notification.

NWDPEventObjectId

Contains the identification of a specific event object

Structure

```
typedef struct {
   NWDPText                name;
   NWDPObjectIdentifier    containingClassOid;
   nint32                  eventType;
} NWDPEventObjectId, N_FAR *pNWDPEventObjectId;
```

Header File

nwdp_att.h

Fields

name

Specifies the object name.

containingClassOid

Specifies the containing object class: "Printer" or "Job."

eventType

Specifies the type of event. The abstract event types are:

1 NWDP_EVENT_TYPE_ERROR
2 NWDP_EVENT_TYPE_WARNING
3 NWDP_EVENT_TYPE_REPORT

The alert event type is:

0 NWDP_EVENT_TYPE_VALUE_CHANGE

Remarks

None

NWDPEventObjectItem

Contains a description of the event object

Structure

```
typedef struct {
   NWDPEventObjectTypeEnum   designator;
   union {
      NWDPObjectIdentifier      filterClassOid;
      NWDPObjectIdentifierSet   eventOidSet;
   } u;
} NWDPEventObjectItem, N_FAR *pNWDPEventObjectItem;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of object:

0 NWDP_EVENT_OBJ_TYPE_OBJECT
1 NWDP_EVENT_OBJ_TYPE_FILTER
2 NWDP_EVENT_OBJ_TYPE_DETAIL

filterClassOid

Specifies the filter for selecting events that are registered for notification. You can filter on such classes as Media Path, Marker, Input, etc.

eventOidSet

Specifies an array of event object identifiers.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union.

NWDPEventObjectSet

Contains a variable length array of NWDPEventObject structures

Structure

```
typedef struct {
   nuint               itemCount;
   pNWDPEventObject    itemPtr;
} NWDPEventObjectSet, N_FAR *pNWDPEventObjectSet;
```

Header File

nwdp_att.h

Fields

itemCount

Specifies the number of elements in the array.

itemPtr

Points to the first element in the contiguous array.

Remarks

To access each element in order, use the following syntax:

```
nint index;
NWDPEventObject array;
for (index = 0; index < array.itemCount; index++)
{
   if (array.itemPtr[index].eventType == 0)
       break;
}
if (index == array.itemCount)
   printf("Not found.\n");
```

NWDPEventReport

Contains information returned as part of event notification

Structure

```
typedef struct {
   NWDPOctetString            supplierId;
   nuint32                    eventType;
   NWDPObjectIdentifier       containingClassOid;
   NWDPObjectIdentification   containingObjectId;
   NWDPObjectIdentifier       filterClassOid;
   NWDPObjectIdentifier       objectClassOid;
   NWDPObjectIdentification   objectId;
   NWDPObjectIdentifier       eventOid;
   NWDPAttribute              eventAttr;
   NWDPText                   message;
   nuint32                    time;
   NWDPQualifiedName          accountOption;
} NWDPEventReport, N_FAR *pNWDPEventReport;
```

Header File

nwdp_nto.h

Fields

supplierId

Specifies the application-defined information which was supplied by the caller when the profile was registered.

eventType

Specifies the type of event. The abstract event types are:

1 NWDP_EVENT_TYPE_ERROR
2 NWDP_EVENT_TYPE_WARNING
3 NWDP_EVENT_TYPE_REPORT

The alert event type is:

0 NWDP_EVENT_TYPE_VALUE_CHANGE

containingClassOid

Specifies the containing object class, either "Printer" or "Job."

containingObjectId

Specifies the object identification of the containing object.

filterClassOid

Specifies the filtering class associated with this event. When dealing

with **NWDPNtfyGetSupportedEvents**, the *attributeID* represents the event and its value is the filter returned here.

objectClassOid

Specifies the class of the object registering the event.

objectId

Specifies the object identification of the object registering the event.

eventOid

Specifies the event which caused the report.

eventAttr

Specifies the new value of the attribute if the event is an alert event (NWDP_EVENT_TYPE_VALUE_CHANGE).

message

Specifies a message generated by the notification server using the information in this event report.

time

Specifies the time of the event as reported by the printer (the Notification supplier).

accountOption

This field is not currently implemented.

Remarks

None

NWDPFinishing

Contains information about a finishing process

Structure

```
typedef struct {
   pNWDPNameOrOid       messageOptionPtr;
   NNWDPFinishingSpec   spec;
} NWDPFinishing, N_FAR *pNWDPFinishing;
```

Header File

nwdp_att.h

Fields

messageOptionPtr

Points to an optional message describing the finishing process.

spec

Specifies the detailed finishing process.

Remarks

NWDPFinishing corresponds to ISO 10175-1: section 9.13.8.

NWDPFinishingProcSpec

Contains a description of a finishing process

Structure

```
typedef struct {
   NWDPFinishingProcEnum   designator;
   union {
      NWDPStitchingSpec         stitchingSpec;
      NWDPBindingSpec           bindingSpec;
      NWDPTrimmingSpec          trimmingSpec;
      NWDPDieCuttingSpec        dieCuttingSpec;
      NWDPPunchingSpec          punchingSpec;
      NWDPPerforatingSpec       perforatingSpec;
      NWDPSlittingSpec          slittingSpec;
      NWDPInsertSpec            insertSpec;
      NWDPCoverSpec             coverSpec;
      NWDPFoldingSpec           foldingSpec;
      NWDPOtherFinishingSpec    otherFinishingSpec;
   } u;
} NWDPFinishingProcSpec, N_FAR *pNWDPFinishingProcSpec;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of finishing process:

0 NWDP_FINISHING_PROC_STITCHING
1 NWDP_FINISHING_PROC_BINDING
2 NWDP_FINISHING_PROC_TRIMMING
3 NWDP_FINISHING_PROC_DIE_CUTTING
4 NWDP_FINISHING_PROC_PUNCHING
5 NWDP_FINISHING_PROC_PERFORATING
6 NWDP_FINISHING_PROC_SLITTING
7 NWDP_FINISHING_PROC_INSERT
8 NWDP_FINISHING_PROC_COVER
9 NWDP_FINISHING_PROC_FOLDING
10 NWDP_FINISHING_PROC_OTHER

stitchingSpec

Specifies a stitching specification for the finishing process.

bindingSpec

Specifies a binding specification for the finishing process.

trimmingSpec

Specifies a trimming specification for the finishing process.

dieCuttingSpec

Specifies a die cutting specification for the finishing process.

punchingSpec

Specifies a punching specification for the finishing process.

perforatingSpec

Specifies a perforating specification for the finishing process.

slittingSpec

Specifies a slitting specification for the finishing process.

insertSpec

Specifies an inserting specification for the finishing process.

coverSpec

Specifies a cover specification for the finishing process.

foldingSpec

Specifies a folding specification for the finishing process.

otherFinishingSpec

Specifies other finishing specifications for the finishing process.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPFinishingProcessSpec corresponds to ISO 10175-1: section 9.13.8.4.

NWDPFinishingProcSpecSeq

Contains a variable length array of NWDPFinishingProcSpec structures

Structure

```
typedef struct {
   nuint                    itemCount;
   pNWDPFinishingProcSpec   itemPtr;
} NWDPFinishingProcSpecSeq, N_FAR *pNWDPFinishingProcSpecSeq;
```

Header File

nwdp_att.h

Fields

itemCount

Specifies the number of elements in the array.

itemPtr

Points to the first element in the contiguous array.

Remarks

To access each element in order, use the following syntax:

```
nint index;
NWDPFinishingProcSpec array;
for (index = 0; index < array.itemCount; index++)
{
   if (array.itemPtr[index].stitchingSpec == 0)
       break;
}
if (index == array.itemCount)
   printf("Not found.\n");
```

NWDPFinishingSpec

Contains information about the type of a finishing specification

Structure

```
typedef struct {
   NWDPFinishingSpecEnum   designator;
   union {
      NWDPNameOrOid              named;
      NWDPFinishingProcSpecSeq   finishingSpecList;
   } u;
} NWDPFinishingSpec, N_FAR *pNWDPFinishingSpec;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of finishing:

0 NWDP_FINISHING_SPEC_NAMED
1 NWDP_FINISHING_SPEC_LIST

named

Specifies a particular finishing object.

finishingSpecList

Specifies a list of finishing processes.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPFinishingSpec corresponds to ISO 10175-1: section 9.13.8.2.

NWDPFoldingParameters

Describes the folding parameters within a finishing specification

Structure

```
typedef struct {
   NWDPCommonParameters   common;
   NWDPLocations          headLocations;
} NWDPFoldingParameters, N_FAR *pNWDPFoldingParameters;
```

Header File

nwdp_att.h

Fields

common

Specifies generic finishing process parameters.

headLocations

Specifies a head location set; corresponds to ISO 10175-1: section 9.13.8.3.7.

Remarks

NWDPFoldingParameters corresponds to ISO 10175-1: section 9.13.8.14.2.

NWDPFoldingSpec

Contains information about the type of folding specification

Structure

```
typedef struct {
   NWDPFoldingSpecEnum        designator;
   union {
      NWDPNameOrOid           namedFolding;
      NWDPFoldingParameters   parameters;
   } u;
} NWDPFoldingSpec, N_FAR *pNWDPFoldingSpec;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of folding specification:

0 NWDP_FOLDING_SPEC_NAMED
1 NWDP_FOLDING_SPEC_PARAMETERS

namedFolding

Specifies a particular folding specification.

parameters

Specifies a set of folding parameters.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPFoldingSpec corresponds to ISO 10175-1: section 9.13.8.14.

NWDPFontSubstitution

Contains information about fonts and their substitutes

Structure

```
typedef struct {
   NWDPText   original;
   NWDPText   substitution;
} NWDPFontSubstitution, N_FAR *pNWDPFontSubstitution;
```

Header File

nwdp_att.h

Fields

original

Specifies the original font.

substitution

Specifies the font to substitute for the original font.

Remarks

NWDPFontSubstitution corresponds to ISO 10175-1: section 9.3.2.11.

NWDPIgnoredAttribute

Contains information about non-compulsory attributes ignored by the server

Structure

```
typedef struct {
   nuint32                  documentNumberOption;
   NWDPObjectIdentifier     attributeId;
   NWDPIgnoredAttrValueSet  values;
} NWDPIgnoredAttribute, N_FAR *pNWDPIgnoredAttribute;
```

Header File

nwdp_att.h

Fields

documentNumberOption

Specifies an optional document sequence number.

attributeId

Specifies the ID of the ignored attribute.

values

Specifies the value of the ignored attribute.

Remarks

NWDPIgnoredAttribute corresponds to ISO 10175-1: section 9.2.8.29.

NWDPIgnoredAttributeValue

Contains the value which was ignored by the Printer/Printer Manager

Structure

```
typedef struct {
   NWDPAVTEnum                  designator;
   union {
      NWDPObjectIdentifier   identifier;
      NWDPNameOrOid          nameOrOid;
   } u;
} NWDPIgnoredAttributeValue, N_FAR *pNWDPIgnoredAttributeValue;
```

Header File

nwdp_att.h

Fields

designator

Specifies the attribute value type:

36 NWDP_AVT_OBJECT_IDENTIFIER
38 NWDP_AVT_NAME_OR_OID

identifier

Specifies the value of the ignored attribute as a single object identifier.

nameOrOid

Specifies the value of the attribute that was ignored as being type Name or OID.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. The NWDPIgnoredAttributeValue structure is designed to support a full range of value types; however, only two types are currently supported: NWDP_AVT_OBJECT_IDENTIFIER and NWDP_AVT_NAME_OR_OID.

NWDPIgnoredAttrValueSet

Contains a variable length array of NWDPIgnoredAttrValue structures

Structure

```
typedef struct {
   nuint                          itemCount;
   pNWDPIgnoredAttributeValue     itemPtr;
} NWDPIgnoredAttrValueSet, N_FAR *pNWDPIgnoredAttrValueSet;
```

Header File

nwdp_att.h

Fields

itemCount

Specifies the number of elements in the array.

itemPtr

Points to the first element in the contiguous array.

Remarks

To access each element in order, use the following syntax:

```
nint index;
NWDPIgnoredAttributeValue array;
for (index = 0; index < array.itemCount; index++)
{
   if (array.itemPtr[index].text == 0)
       break;
}
if (index == array.itemCount)
   printf("Not found.\n");
```

NWDPImposition

Contains information about logical page placement

Structure

```
typedef struct {
   NWDPImpositionEnum   designator;
   union {
      NWDPNameOrOid        impositionObject;
      NWDPImpositionAttr   impositionAttr;
   } u;
} NWDPImposition, N_FAR *pNWDPImposition;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of imposition:

0 NWDP_IMPOSITION_OBJECT
1 NWDP_IMPOSITION_ATTR

impositionObject

Specifies an imposition object that describes the placement of multiple logical page images onto a single physical page; corresponds to ISO 10175-1: section 6.3.14.

impositionAttr

Specifies an imposition attribute list.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPImposition corresponds to ISO 10175-1: section 6.3.14.

NWDPImpositionAttr

Contain information about an imposition type

Structure

```
typedef struct {
   NWDPImpositionAttrEnum   designator;
   union {
      struct {
         nuint    itemCount;
         pnint8   itemPtr;
      } plex;
      nint32    numberUp;
      nreal64   xImageShift;
      nreal64   yImageShift;
      struct {
         nuint    itemCount;
         pnint8   itemPtr;
      } logPageOrigin;
      nreal64   logPageXOffset;
      nreal64   logPageYOffset;
      nreal64   logPageScaling;
   } u;
} NWDPImpositionAttr, N_FAR *pNWDPImpositionAttr;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of imposition attribute:

0 NWDP_IMP_ATTR_PLEX
1 NWDP_IMP_ATTR_NUMBER_UP
2 NWDP_IMP_ATTR_X_IMAGE_SHIFT
3 NWDP_IMP_ATTR_Y_IMAGE_SHIFT
4 NWDP_IMP_ATTR_LOG_PAGE_ORIGIN
5 NWDP_IMP_ATTR_LOG_PAGE_X_OFFSET
6 NWDP_IMP_ATTR_LOG_PAGE_Y_OFFSET
7 NWDP_IMP_ATTR_LOG_PAGE_SCALING

itemCount

Specifies the number of array elements or items.

itemPtr

Points to the first element in the contiguous array.

numberUp

Specifies the number of source page images to impose on a single instance of a selected medium; corresponds to ISO 10175-1: section 9.3.2.16.1.

xImageShift

Specifies the X-axis shift.

yImageShift

Specifies the Y-axis shift.

itemCount

Specifies the number of array elements or items.

itemPtr

Points to the first element in the contiguous array.

logPageXOffset

Specifies the distance that one of the logical pages is to shifted parallel to the x-axis of its respective Logical Page Coordinate System (LPCS); corresponds to ISO 10175-1: section 9.15.4.

logPageYOffset

Specifies the distance that one of the logical pages is to shifted parallel to the y-axis of its respective Logical Page Coordinate System (LPCS); corresponds to ISO 10175-1: section 9.15.5.

logPageScaling

Specifies the scale factors to be applied to transform the respective source pages into their corresponding logical page sizes; corresponds to ISO 10175-1: section 9.15.6

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPImpositionAttr corresponds to ISO 10175-1: section 9.15

NWDPInfoItem

Contains information about an item's type

Structure

```
typedef struct {
   NWDPInfoItemEnum   designator;
   union {
      nuint8            uInt8;
      nuint16           uInt16;
      nuint32           uInt32;
      nbool             flag;
      pnstr             nString;
      NWDPOctetString   octetString;
   } u;
} NWDPInfoItem, N_FAR *pNWDPInfoItem;
```

Header File

nwdp_srs.h

Fields

designator

Specifies the type of information:

0 NWDP_INFO_ITEM_INT8
1 NWDP_INFO_ITEM_INT16
2 NWDP_INFO_ITEM_INT32
3 NWDP_INFO_ITEM_BOOL
4 NWDP_INFO_ITEM_STRING
5 NWDP_INFO_ITEM_OCTET_STR

uInt8

Specifies an unsigned 8-bit integer value.

uInt16

Specifies an unsigned 16-bit integer value.

uInt32

Specifies an unsigned 32-bit integer value.

flag

Specifies a Boolean value.

nString

Specifies an ASCII null-terminated string value.

octetString

Specifies an octet string value.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union.

NWDPInputTrayMedium

Contains information about the medium in each input tray of the printer

Structure

```
typedef struct {
   NWDPNameOrOid   inputTray;
   NWDPNameOrOid   medium;
} NWDPInputTrayMedium, N_FAR *pNWDPInputTrayMedium;
```

Header File

nwdp_att.h

Fields

inputTray

Specifies the input tray; corresponds to ISO 10175-1: section 9.4.32.

medium

Specifies the medium in the selected input tray; corresponds to ISO 10175-1: section 9.6.1.

Remarks

NWDPInputTrayMedium corresponds to ISO 10175-1: section 9.4.34.

NWDPInsertId

Contains information about an insert sheet to be used as part of the finishing process

Structure

```
typedef struct {
   NWDPInsertIdEnum   designator;
   union {
      NWDPNameOrOid   insertName;
      nuint32         insertBin;
   } u;
} NWDPInsertId, N_FAR *pNWDPInsertId;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of insert identifier:

0 NWDP_INSERT_NAME
1 NWDP_INSERT_BIN

insertName

Specifies the name of an insert sheet to be inserted into the document corresponds to ISO 10175-1: section 9.13.8.12.6.

insertBin

Specifies the bin from which the insert sheet can be obtained by the inserting finishing process; corresponds to ISO 10175-1: section 9.13.8.12.7.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPInsertId corresponds to ISO 10175-1: section 9.13.8.12.5.

NWDPInsertParameters

Contains parameters for inserting a sheet during a finishing process

Structure

```
typedef struct {
   NWDPEdgeOption       referenceEdgeOption;
   NWDPEdgeOption       jogEdgeOption;
   NWDPInsertSheetSeq   insertSheetList;
} NWDPInsertParameters, N_FAR *pNWDPInsertParameters;
```

Header File

nwdp_att.h

Fields

referenceEdgeOption

Specifies an optional reference edge value; corresponds to ISO 10175-1: section 9.13.8.12.2.

jogEdgeOption

Specifies an optional jog edge option; corresponds to ISO 10175-1: section 9.13.8.12.2.

insertSheetList

Specifies a list of insert sheets that specify the set of properties which identify a sheet to be inserted in the document by the inserting finishing process; corresponds to ISO 10175-1: section 9.13.8.12.3.

Remarks

NWDPInsertParameters corresponds to ISO 10175-1: section 9.4.32.

NWDPInsertSheet

Contains a set of properties that identify a sheet to be inserted in a document by an insert finishing process

Structure

```
typedef struct {
   NWDPInsertId               insertId;
   pNWDPXYDimensions          insertSizeOptionPtr;
   NWDPEdgeOption             insertEdgeOption;
   NWDPInsertTopSurfaceEnum   insertTopSurface;
   nuint32                    insertAfter;
   pNWDPNameOrOid             insertMessageOptionPtr;
} NWDPInsertSheet, N_FAR *pNWDPInsertSheet;
```

Header File

nwdp_att.h

Fields

insertId

Specifies the insert sheet.

insertSizeOptionPtr

Specifies a size for the insert. (Optional)

insertEdgeOption

Specifies an edge for the insert. (Optional)

insertTopSurface

Specifies the type of top surface for the insert; corresponds to ISO 10175-1: section 9.13.8.12.8:

0 NWDP_INSERT_TOP_SURFACE_TOP
1 NWDP_INSERT_TOP_SURFACE_BOTTOM

insertAfter

Specifies the ordinal number of the document page after which the insert sheet should be inserted; corresponds to ISO 10175-1: section 9.13.8.12.9.

insertMessageOptionPtr

Specifies a message which might describe the insert sheet and other information; corresponds to ISO 10175-1: section 9.13.8.12.10. (Optional)

Remarks

NWDPInsertSheet corresponds to ISO 10175-1: section 9.13.8.12.4.

NWDPInsertSheetSeq

Contains a variable length array of NWDPInsertSheet structures

Structure

```
typedef struct {
   nuint               itemCount;
   pNWDPInsertSheet    itemPtr;
} NWDPInsertSheetSeq, N_FAR *pNWDPInsertSheetSeq;
```

Header File

nwdp_att.h

Fields

itemCount

Specifies the number of elements in the array.

itemPtr

Points to the first element in the contiguous array.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union.

To access each element in order, use the following syntax:

```
nint index;
NWDPInsertSheet array;
for (index = 0; index < array.itemCount; index++)
{
   if (array.itemPtr[index].insertId == 0)
       break;
}
if (index == array.itemCount)
   printf("Not found.\n");
```

NWDPInsertSpec

Contains information about the type of an insert specification

Structure

```
typedef struct {
   NWDPInsertSpecEnum   designator;
   union {
      NWDPNameOrOid          namedInserting;
      NWDPInsertParameters   parameters;
   } u;
} NWDPInsertSpec, N_FAR *pNWDPInsertSpec;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of insert specification:

0 NWDP_INSERT_SPEC_NAMED
1 NWDP_INSERT_SPEC_PARAMETERS

namedInserting

Specifies a named insert specification.

parameters

Specifies an inserting parameter set.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPInsertSpec corresponds to ISO 10175-1: section 9.13.8.12.

NWDPIntegerOption

Contains an integer value or an indicator that no value is present

Structure

```
typedef struct {
   nint32   flag;
   nint32   value;
} NWDPIntegerOption, N_FAR *pNWDPIntegerOption;
```

Header File

nwdp_att.h

Fields

flag

Specifies whether the value of the ID is valid. *flag* is a boolean value which specifies whether the value field of the optional integer is present. Zero means there is no integer.

value

Specifies a 32-bit integer (only valid if *flag* is non-zero).

Remarks

None

NWDPIntegerRange

Describes a range of integer values

Structure

```
typedef struct {
   nint32   lowerBound;
   nint32   upperBound;
} NWDPIntegerRange, N_FAR *pNWDPIntegerRange;
```

Header File

nwdp_att.h

Fields

lowerBound

Specifies the lower boundary of the value range.

upperBound

Specifies the upper boundary of the value range.

Remarks

NWDPIntegerRange corresponds with ISO 10175-1: section 9.1.5.16.

NWDPIntegerSeq

Contains a variable length array of integer values

Structure

```
typedef struct {
   nuint     itemCount;
   pnint32   itemPtr;
} NWDPIntegerSeq, N_FAR *pNWDPIntegerSeq;
```

Header File

nwdp_att.h

Fields

itemCount

Specifies the number of elements in the array.

itemPtr

Points to the first element in the contiguous array.

Remarks

NWDPIntegerSeq corresponds to ISO 10175-1: section 9.1.5.12.

To access each element in order, use the following syntax:

```
nint index;
NWDPIntegerSeq array;
for (index = 0; index < array.itemCount; index++)
{
   if (array.itemPtr[index].lowerbound == 0)
       break;
}
if (index == array.itemCount)
   printf("Not found.\n");
```

NWDPJobCategories

Contains the job's organization and the security integrity categories associated with the organization

Structure

```
typedef struct {
   NWDPObjectIdentifier   organizationId;
   NWDPCardinalSeq        categories;
} NWDPJobCategories, N_FAR *pNWDPJobCategories;
```

Header File

nwdp_att.h

Fields

organizationId

Specifies the organization.

categories

Specifies the job's security integrity categories.

Remarks

NWDPJobCategories corresponds to ISO 10175-1: section 9.2.7.5.

NWDPJobDocListItem

Returns information on the document

Structure

```
typedef struct
{
   nuint     sequenceNumber;
   pnstr16   docName16Ptr;
   nuint32   copyCount;
   nuint32   docSize;
} NWDPJobDocListItem, N_FAR *pNWDPJobDocListItem;
```

Header File

nwdp_job.h

Fields

sequenceNumber

Specifies the document's sequence number.

docuName16Ptr

Points to the name of the document.

copyCount

Specifies the number of copies to print.

docSize

Specifies the document size.

Remarks

NWDPJobDocListItem corresponds to ISO 10175-1: section 9.3.5.

NWDPJobLevel

Contains the job's security integrity level information

Structure

```
typedef struct {
   NWDPObjectIdentifier   organizationId;
   nuint32                level;
} NWDPJobLevel, N_FAR *NWDPJobLevel;
```

Header File

nwdp_att.h

Fields

organizationId

Specifies the organization.

level

Specifies the security integrity level.

Remarks

NWDPJobLevel corresponds to ISO 10175-1: section 9.2.7.4.

NWDPLevelRange

Contains a range of job integrity level values

Structure

```
typedef struct {
   NWDPObjectIdentifier   organizationId;
   nint32                 minimum;
   nint32                 maximum;
} NWDPLevelRange, N_FAR *pNWDPLevelRange;
```

Header File

nwdp_att.h

Fields

organizationId

Specifies the organization.

minimum

Specifies the minimum security integrity level allowable to print the job.

maximum

Specifies the maximum security integrity level allowable to print the job.

Remarks

NWDPLevelRange corresponds to part of the id-val-integrity-policy-biba data from ISO 10175-1: section 9.2.7.6.

NWDPLocations

Describes distances from a reference edge

Structure

```
typedef struct {
   NWDPLocationValue   value;
   NWDPRealOption      toleranceOption;
} NWDPLocations, N_FAR *pNWDPLocations;
```

Header File

nwdp_att.h

Fields

value

Specifies a distance from a reference edge in millimeters.

toleranceOption

Specifies an optional tolerance in millimeters.

Remarks

NWDPLocations corresponds to ISO 10175-1: section 9.1.5.46.

NWDPLocationValue

Contains information about the type of a location

Structure

```
typedef struct {
   NWDPLocationiValueEnum   designator;
   union {
      NWDPRealSet     numericValueSeq;
      NWDPNameOrOid   namedValue;
   } u;
} NWDPLocationValue, N_FAR *pNWDPLocationValue;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of location:

0 NWDP_LOCATION_NUMERIC
1 NWDP_LOCATION_NAMED

numericValueSeq

Specifies a sequence of location values.

namedValue

Specifies a named location.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPLocationValue corresponds to ISO 10175-1: section 9.1.5.46.

NWDPMediaSelect

Contains specifications for overriding the default medium for specified pages

Structure

```
typedef struct {
   NWDPPageSelect   pageRange;
   NWDPNameOrOid    mediumId;
} NWDPMediaSelect, N_FAR *pNWDPMediaSelect;
```

Header File

nwdp_att.h

Fields

pageRange

Specifies the page range, for example pages. 5-23.

mediumId

Specifies the medium to be used for the specified pages.

Remarks

This is part of the mechanism that allows, for example, the first page of a letter to be on letterhead and subsequent pages on plain paper.

NWDPMediaSelectSeq

Contains a variable length array of NWDPMediaSelect structures

Structure

```
typedef struct {
   nuint               itemCount;
   pNWDPMediaSelect    itemPtr;
} NWDPMediaSelectSeq, N_FAR *pNWDPMediaSelectSeq;
```

Header File

nwdp_att.h

Fields

itemCount

Specifies the number of elements in the array.

itemPtr

Points to the first item in the contiguous array.

Remarks

To access each element in order, use the following syntax:

```
nint index;
NWDPMediaSelect array;
for (index = 0; index < array.itemCount; index++)
{
   if (array.itemPtr[index].pageRange == 0)
       break;
}
if (index == array.itemCount)
   printf("Not found.\n");
```

NWDPMediumContinuousSizes

Contains sizes for the input trays when the trays are adjustable

Structure

```
typedef struct {
   NWDPRealRange   sizeRangeAcrossFeedDirection;
   nreal64         sizeIncrementAcrossFeedDir;
   NWDPRealRange   sizeRangeInFeedDirection;
   nreal64         sizeIncrementInFeedDir;
   nbool           longEdgeFeeds;
   NWDPArea        GenericAssuredReproductionArea;
} NWDPMediumContinuousSizes, N_FAR *pNWDPMediumContinuousSizes;
```

Header File

nwdp_att.h

Fields

sizeRangeAcrossFeedDirection

Specifies the size range across the direction of the paper feed.

sizeIncrementAcrossFeedDir

Specifies the spacing between adjustment positions across the feed direction.

sizeRangeInFeedDirection

Specifies the size range in the direction of the paper feed.

sizeIncrementInFeedDir

Specifies the spacing between adjustment positions in the feed direction.

longEdgeFeeds

Specifies a value of TRUE if the paper's long edge is fed into the printer from the input tray.

GenericAssuredReproductionArea

Specifies the area of the paper where it is possible to print.

Remarks

NWDPMediumContinuousSizes corresponds to ISO 10175-1:section 9.4.16.

NWDPMediumDiscreteSizes

Contains the size for an input tray when the tray is not adjustable

Structure

```
typedef struct {
   NWDPPageSize   pageSize;
   nbool          longEdgeFeeds;
   NWDPArea       assuredReproductionArea;
} NWDPMediumDiscreteSizes, N_FAR *pNWDPMediumDiscreteSizes;
```

Header File

nwdp_att.h

Fields

pageSize

Specifies the size of the page that is contained in the tray.

longEdgeFeeds

Specifies a value of TRUE if the paper's long edge is fed into the printer from the input tray.

assuredReproductionArea

Specifies the area of the paper where it is possible to print.

Remarks

NWDPMediumDiscreteSizes corresponds to ISO 10175-1:section 9.4.16.

NWDPMediumSize

Contains a description of the size(s) that the input tray can handle

Structure

```
typedef struct {
   NWDPMediumSizeEnum   designator;
   union {
      NWDPMediumDiscreteSizes     discreteSizes;
      NWDPMediumContinuousSizes   continuousSizes;
   } u;
} NWDPMediumSize, N_FAR *pNWDPMediumSize;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of medium size:

0 NWDP_MEDIUM_SIZE_DISCRETE
1 NWDP_MEDIUM_SIZE_CONTINUOUS

discreteSizes

Specifies the description of a single sized tray.

continuousSizes

Specifies the description of an adjustable sized tray.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPMediumSize corresponds to ISO 10175-1:section 9.4.16.

NWDPMediumSourceSize

Contains a description of the input medium size and an optional tray location

Structure

```
typedef struct {
   pNWDPNameOrOid   inputTrayOptionPtr;
   NWDPMediumSize   mediumSize;
} NWDPMediumSourceSize, N_FAR *pNWDPMediumSourceSize;
```

Header File

nwdp_att.h

Fields

inputTrayOptionPtr

Points to a description of a tray which contains media of this size. (Optional)

mediumSize

Specifies the dimensions, fixed or adjustable, of the medium, its feed direction and its reproduction area.

Remarks

NWDPMediumSourceSize corresponds to ISO 10175-1:section 9.4.16.

NWDPMediumSubstitution

Contains a substitution of one medium for another allowed for the document in which it is found

Structure

```
typedef struct {
   NWDPNameOrOid   original;
   NWDPNameOrOid   substitution;
} NWDPMediumSubstitution, N_FAR *pNWDPMediumSubstitution;
```

Header File

nwdp_att.h

Fields

original

Specifies the original medium.

substitution

Specifies the allowable substitute medium.

Remarks

NWDPMediumSubstitution corresponds to ISO 10175-1:section 9.3.2.9.

NWDPNameOrOid

Contains either a user-supplied name or a globally known OID used to identify something

Structure

```
typedef struct {
   NWDPNameOrOidEnum   designator;
   union {
      NWDPObjectIdentifier   globalForm;
      NWDPText               localForm;
   } u;
} NWDPNameOrOid, N_FAR *pNWDPNameOrOid;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of name or OID:

0 NWDP_NAME_OR_OID_GLOBAL
1 NWDP_NAME_OR_OID_LOCAL

globalForm

Specifies an OID.

localForm

Specifies a Unicode name.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPNameOrOid corresponds to ISO 10175-1:section 9.1.5.38.

NWDPNameOrOidDimensionMap

Contains a pairing of a name or Oid and a set of integers used to describe dimensions

Structure

```
typedef struct {
   NWDPNameOrOid              nameOrOid;
   NWDPXYCardinalDimensions   value;
} NWDPNameOrOidDimensionMap, N_FAR *pNWDPNameOrOidDimensionMap;
```

Header File

nwdp_att.h

Fields

nameOrOid

Specifies the name of the dimensions pair.

value

Specifies the values for the X and Y dimensions.

Remarks

None

NWDPNameOrOidSet

Contains a variable length array of NWDPNameOrOid structures

Structure

```
typedef struct {
   nuint             itemCount;
   pNWDPNameOrOid    itemPtr;
} NWDPNameOrOidSet, N_FAR *pNWDPNameOrOidSet;
```

Header File

nwdp_att.h

Fields

itemCount

Specifies the number of elements in the array.

itemPtr

Points to the first element in the array.

Remarks

NWDPNameOrOidSet corresponds to ISO 10175-1: section 9.1.5.39.

To access each element in order, use the following syntax:

```
nint index;
NWDPNameOrOid array;
for (index = 0; index < array.itemCount; index++)
{
   if (array.itemPtr[index].nameOrOid == 0)
       break;
}
if (index == array.itemCount)
   printf("Not found.\n");
```

NWDPNdpsCred0

Contains a server and connection on that server used for authentication of a printer client

Structure

```
typedef struct {
   pnstr     serverNamePtr;
   nuint16   connection;
} NWDPNdpsCred0, N_FAR *pNWDPNdpsCred0;
```

Header File

nwdp_att.h

Fields

serverNamePtr

Points to the SAP name of the server that the client is logged into. Consists of 2-47 8-bit characters.

connection

Specifies the connection number (not handle) on the server.

Remarks

This credential allows validation of clients that are logged into the specified server. The NDS tree to which the server belongs is the only tree for which the caller will have credentials.

NWDPNdsName

Contains the Unicode canonicalized distinguished name and optionally the Unicode tree name for an NDS object.

Structure

```
typedef struct {
   NWDPText   object;
   NWDPText   treeOption;
} NWDPNdsName, N_FAR *pNWDPNdsName;
```

Header File

nwdp_att.h

Fields

object

Specifies the null-terminated, canonicalized, Unicode distinguished name for the NDS object.

treeOption

Specifies a null-terminated Unicode tree name. If omitted, the current tree name is assumed. (Optional)

Remarks

The object name can be specified relative to the current context, but when sending name data to another entity on the tree, the full name is required.

NWDPNetAddress

Contains information about a network address

Structure

```
typedef struct {
   NWDPNetAddressTypeEnum   type;
   NWDPOctetString          address;
} NWDPNetAddress, N_FAR *pNWDPNetAddress;
```

Header File

nwdp_att.h

Fields

type

Specifies the net address topology type:

0 NWDP_NET_IPX
1 NWDP_NET_IP
2 NWDP_NET_SDLC
3 NWDP_NET_TOKENRING_ETHERNET
4 NWDP_NET_OSI
5 NWDP_NET_APPLETALK
6 NWDP_NET_COUNT

address

Specifies the network address.

Remarks

The network address is specified as an octet string and a type which is used to identify the syntax of the octet string.

NWDPNotifyDeliveryAddr

Contains notification delivery addresses

Structure

```
typedef struct {
   nuint              itemCount;
   pNWDPAddressItem   itemPtr;
} NWDPNotifyDeliveryAddr, N_FAR *pNWDPNotifyDeliveryAddr;
```

Header File

nwdp_att.h

Fields

itemCount

Specifies the number of destination delivery addresses for the notifications.

itemPtr

Points to the first of a block ofNWDPAddressItem structures, which tell the notification service where to send the notifications.

Remarks

None

NWDPNSrvFQN

Contains a Name Service fully qualified name

Structure

```
typedef struct {
   NWDPNSrvSyntaxEnum   designator;
   union {
      struct {
         pnstr16   dn16Ptr;
         pnstr16   treeOption16Ptr;
      } ndsWithTree;
   } u;
} NWDPNSrvFQN, N_FAR *pNWDPNSrvFQN;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of qualified name:

0 NWDP_NSRV_SYNTAX_NDS_WITH_TREE

dn16Ptr

Specifies the fully canonicalized, null-terminated Unicode NDS object name with a leading period.

treeOption16Ptr

Points to the tree name where the *dn16Ptr* object name is valid. (Optional)

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. This structure is defined as an NDS object name, fully canonicalized and converted to Unicode. This name is qualified by the tree name. The tree, if omitted, is implicitly the one contained in the NDS context identified by the call to **NWDPNSrvSetNativeNDSContext**.

NWDPNSrvFQNBuffer

Contains an NWDPNSrvFQN structure and sufficient space for worst case data

Structure

```
typedef struct {
   NWDPNSrvFQN   fqn;
   union {
      nchar16 ndsWithTree[NWDP_NSRV_MAX_DN_CHARS +
                          NWDP_NSRV_MAX_TREE_CHARS+2];
   } bufferData;
   } u;
} NWDPNSrvFQNBuffer, N_FAR *pNWDPNSrvFQNBuffer;
```

Header File

nwdp_att.h

Fields

fqn

Specifies the pointers for the NDS distinguished object name and tree name.

ndsWithTree

Specifies the worst case usage for a Unicode distinguished name (256 characters), a Unicode tree name (32 characters), and two terminating NULL characters.

Remarks

The API uses *pNWDPNSrvFQNBuffer* to indicate that such a worst case buffer should be used. The *fqn* substructure is initialized to point within the buffer space when such data is returned through the interface.

NWDPNSrvObjListItem

Returns the name service object name and its class

Structure

```
typedef struct
{
   NWDPNSrvObjType   objectType;
   nchar             objectName16[ NWDP_NSRV_OBJECT_NAME_CHARS + 1];
} NWDPNSrvObjListItem, N_FAR *pNWDPNSrvObjListItem;
```

Header File

nwdp_nsr.h

Fields

objectType

Specifies the class of the object.

objectName16

Specifies the name of the object.

Remarks

None

NWDPNtfyDeliveryMethod

Contains information about the delivery method used for event notification

Structure

```
typedef struct {
   NWDPNameOrOid   methodId;
   pnstr16         methodName16Ptr;
   pnstr16         methodVersion16Ptr;
   pnstr16         fileName16Ptr;
   nbool           adminSubmitProfileFlag;
} NWDPNtfyDeliveryMethod, N_FAR *pNWDPNtfyDeliveryMethod;
```

Header File

nwdp_ntt.h

Fields

methodId

Specifies the OID representing the delivery method ID.

methodName16Ptr

Points to the delivery method name.

methodVersion16Ptr

Points to the delivery method version number.

fileName16Ptr

Points to the delivery method file name, which is the name of the file used to load it on the server.

adminSubmitProfileFlag

Specifies the flag used by the NwAdmin utility. The delivery method sets this flag based on whether it makes sense for an administrator to enable notification for someone else using this delivery method.

Remarks

None

NWDPNtfyEventObjectListItem

Contains description of an event object

Structure

```
typedef struct {
   nuint32                      eventType;
   pNWDPOid                     containingClassOidPtr;
   pNWDPObjectIdentification    containingObjectIdPtr;
   NWDPEventObjectTypeEnum      designator;
   union {
      pNWDPOid         filterClassOidPtr;
      NWDPOidSetRef    eventOidSetRef;
   } u;
} NWDPNtfyEventObjectListItem, N_FAR *pNWDPNtfyEventObjectListItem;
```

Header File

nwdp_not.h

Fields

eventType

Specifies the type of event. The abstract event types are:

1 NWDP_EVENT_TYPE_ERROR
2 NWDP_EVENT_TYPE_WARNING
3 NWDP_EVENT_TYPE_REPORT

The alert event type is:

0 NWDP_EVENT_TYPE_VALUE_CHANGE

containingClassOidPtr

Points to the containing object class, either "Printer" or "Job," which is the same class returned in NWDPEventObjectId when calling NWDPNtfyGetSupportedEvents.

containingObjectIdPtr

Points to the object whose events are being monitored for notification.

designator

Specifies the type of object:

0 NWDP_EVENT_OBJ_TYPE_OBJECT

This is an event object. The union is not used and notification is sent for all events occurring on the object designated by the *containingObjectIdPrt* field that match the *eventType* field.

2 NWDP_EVENT_OBJ_TYPE_DETAIL

This is a detail event object and the *eventOidSetRef* field contains the list of events.

1 NWDP_EVENT_OBJ_TYPE_FILTER

This is a filter event and the *filterClassOidPtr* field contains the filter.

filterClassOidPtr

Points to a filter for selecting events. This is a smaller set than the class in the *containgClassOidPtr* field.

eventOidSetRef

Specifies the events by a set of OIDs.

Remarks

The library fills out and returns the NWDPNtfyEventObjectListitem structure when you call the **NWDPNtfyListEventObjects** function.

NWDPNtfyGetSupportedListItem

Contains events for which notification can be requested by a given printer

Structure

```
typedef struct {
   pNWDPEventObjectId   eventObjectIdPtr;
   NWDPAttrSetRef       eventAttrSetRef;
   NWDPASAVPRef         eventAvpRef;
} NWDPNtfyGetSupportedListItem, N_FAR *pNWDPNtfyGetSupportedListItem;
```

Header File

nwdp_not.h

Fields

eventObjectIdPtr

Identifies the printer agent name, the containing class (for example, "Job"), and the event type (for example, "Error").

eventAttrSetRef

The reference to the event attribute set which contains the events.

eventAvpRef

The attribute-value pointer (avp) is used to traverse the attribute set referenced as *eventAttrSetRef*.

Remarks

The library fills out and returns the NWDPNtfyGetSupportedListItem structure when you call the **NWDPGetSupportedEvents** function.

NWDPNtfyProfileInfo

Contains information needed to create a profile reference

Structure

```
typedef struct {
   NWDPPersistenceEnum      persistence;
   NWDPQualifiedName        consumerName;
   NWDPLanguageId           languageId;
   NWDPNameOrOid            methodId;
   NWDPNotifyDeliveryAddr   deliveryAddress;
   NWDPOctetString          supplierIdOption;
   pNWDPQualifiedName       accountOptionPtr;
} NWDPNtfyProfileInfo, N_FAR *pNWDPNtfyProfileInfo;
```

Header File

nwdp_not.h

Fields

persistence

Specifies the type of persistence:

0 NWDP_PERSISTENCE_PERMANENT
1 NWDP_PERSISTENCE_VOLATILE

consumerName

Specifies the owner of the profile.

languageId

Specifies the language used for notifications. If NWDP_LID_DEFAULT is used, notifications will use the language identified by the environment variable NWLANGUAGE.

methodId

Specifies the method used by the notification service.

deliveryAddress

Specifies the locations where notifications will be sent. Use **NWDPNtfyListPrompts** to determine how to fill in the NWDPAddressItem structure for a given method.

supplierIdOption

Specifies an application-defined parameter, in which the caller can place information to store in the profile and return when the profile is retrieved.

accountOptionPtr

This field is reserved for future use. Set the designator to NWDP_QUALIFIED_NAME_NONE.

Remarks

Set the *persistence* field to NWDP_PERSISTENCE_PERMANENT if you want your profiles to remain in effect when the broker is downed and restarted; otherwise set this field to NWDP_PERSISTENCE_VOLATILE.

NWDPNtfyPromptListItem

Contains information about the prompts used for a notification method

Structure

```
typedef struct {
   nchar16                prompt16[NWDP_NTFY_MAX_PROMPT_CHARS+1];
   NWDPDeliveryAddrItem   addressItem;
} NWDPNtfyPromptListItem, N_FAR *pNWDPNtfyPromptListItem;
```

Header File

nwdp_ntt.h

Fields

prompt16

Specifies the Unicode prompt that give directions on how to fill out the NWDPNotifyDeliveryAddr structure.

addressItem

Specifies restrictions for addresses.

Remarks

If the *addressItem* field is of type NWDP_ADDR_ITEM_USR_OR_CONTAINER, the "userOrContainerRestrict" could be the minimum or maximum lengths of DS names. The *addressItem* field is also contains the value to use for filling the NWDPNotifyDeliveryAddr when adding profiles.

The library fills out and returns the NWDPNtfyPromptListItem structure when you call the **NWDPNtfyListPrompts** function.

NWDPNumbersUpSupported

Contains the number-up values and imposition objects supported by the printer

Structure

```
typedef struct {
   NWDPNumbersUpEnum   designator;
   union {
      nuint32             cardinal;
      NWDPNameOrOid       nameOrOid;
      NWDPCardinalRange   cardinalRange;
   } u;
} NWDPNumbersUpSupported, N_FAR *pNWDPNumbersUpSupported;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of numbers up value:

0 NWDP_NUMBERS_UP_CARDINAL
1 NWDP_NUMBERS_UP_NAME_OR_OID
2 NWDP_NUMBERS_UP_CARDINAL_RANGE

cardinal

Specifies a single number-up.

nameOrOid

Specifies a named set of number-up values to be supported.

cardinalRange

Specifies a range of number-up values; for example, 1-up through 2-up.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPNumberUpSupported corresponds to ISO 10175-1, section 9.4.62.2.

NWDPObjectIdentification

Contains a union of all of the different object identifications supported

Structure

```
typedef struct {
   NWDPObjectIdentificationEnum designator;
   union {
      NWDPPrtContainedObjectId      prtContainedObjectId;
      NWDPDocumentIdentifier        documentIdentifier;
      NWDPObjectIdentifier          objectIdentifier;
      NWDPDistinguishedNameString   objectName;
      NWDPNameOrOid                 nameOrOid;
      NWDPText                      simpleName;
      NWDPPrtConfigObjectId         prtConfigObjectId;
      NWDPQualifiedName             qualifiedName;
      NWDPEventObjectId             eventObjectId;
   } u;
} NWDPObjectIdentification, N_FAR *pNWDPObjectIdentification;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of object ID:

0 NWDP_OBJ_ID_NONE
1 NWDP_OBJ_ID_PRT_CONTAIND_OBJ_ID
2 NWDP_OBJ_ID_DOCUMENT_IDENTIFIER
3 NWDP_OBJ_ID_OBJECT_IDENTIFIER
4 NWDP_OBJ_ID_OBJECT_NAME
5 NWDP_OBJ_ID_NAME_OR_OID
6 NWDP_OBJ_ID_SIMPLE_NAME
7 NWDP_OBJ_ID_PRT_CONFIG_OBJ_ID
8 NWDP_OBJ_ID_QUALIFIED_NAME
9 NWDP_OBJ_ID_EVENT_OBJECT_ID

prtContainedObjectId

Specifies an object contained by a logical printer.

documentIdentifier

Specifies a document.

objectIdentifier

Specifies an object.

objectName

Specifies the name of the object.

nameOrOid

Specifies an identifier that might have either a global or local form (see ISO10175-1: section 9.1.5.38).

simpleName

Specifies a short human readable string representation of a simple name for an object created by an administrator (see ISO10175-1: section 9.1.5.6).

prtConfigObjectId

Specifies a printer configuration object that describes defaults and limits.

qualifiedName

Specifies an object's name, either as a simple name (Unicode zero-terminated string) or as a fully canonicalized NDS name with corresponding NDS Tree name.

eventObjectId

Specifies the identification of an NDPS event. This includes the event type, the name of the event, and the OID of the Containing Class.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. Each of these fields specifies the kind of object mentioned in the naming. Grouping these fields together allows List-Object-Attributes operations (LOAs) to be performed on any object of these types without increasing the complexity of the interface elsewhere. NWDPObjectIdentification corresponds to ISO 10175-1, section 8.2.4.1.

NWDPObjectIdentificationSeq

Contains a set of NWDPObjectIdentification structures

Structure

```
typedef struct {
   nuint                         itemCount;
   pNWDPObjectIdentification     itemPtr;
} NWDPObjectIdentificationSeq, N_FAR *pNWDPObjectIdentificationSeq;
```

Header File

nwdp_att.h

Fields

itemCount

Specifies the number of array elements or items.

itemPtr

Points to the first element in the array.

Remarks

To access each element in order, use the following syntax:

```
nint index;
NWDPObjectIdentification array;
for (index = 0; index < array.itemCount; index++)
{
   if (array.itemPtr[index].designator == 0)
       break;
}
if (index == array.itemCount)
   printf("Not found.\n");
```

NWDPObjectIdentifier

Encapsulates an OID

Structure

```
typedef struct
{
   nuint      oidStructSize;
   pNWDPOid   oidStructPtr;
} NWDPObjectIdentifier, N_FAR *pNWDPObjectIdentifier;
```

Header File

nwdp_att.h

Fields

oidStructSize

Specifies the number of bytes of NWDPOid pointed to by *oidStructPtr*.

oidStructPtr

Points to NWDPOid containing the binary image of an ASN1 object identifier.

NWDPObjectIdentifierSet

Contains a variable length array of NWDPObjectIdentifier structures

Structure

```
typedef struct {
   nuint                    itemCount;
   pNWDPObjectIdentifier    itemPtr;
} NWDPObjectIdentifierSet, N_FAR *pNWDPObjectIdentifierSet;
```

Header File

nwdp_att.h

Fields

itemCount

Specifies the number of elements in the array.

itemPtr

Points to the first element in the array.

Remarks

NWDPObjectIdentifierSet corresponds to ISO 10175-1: section 9.1.5.37.

To access each element in order, use the following syntax:

```
nint index;
NWDPObjectIdentifier array;
for (index = 0; index < array.itemCount; index++)
{
   if (array.itemPtr[index].OidStructSize == 0)
       break;
}
if (index == array.itemCount)
   printf("Not found.\n");
```

NWDPOctetString

Contains a string of bytes and the count of those bytes

Structure

```
typedef struct {
   nuint    itemCount;
   pnint8   itemPtr;
} NWDPOctetString, N_FAR *pNWDPOctetString;
```

Header File

nwdp_att.h

Fields

itemCount

Specifies the number of elements in the array.

itemPtr

Points to the first array element.

NWDPOid

Defines an OID

Structure

```
typedef struct
{
   nuint8   asn1Type;
   nuint8   oidLength;
   nuint8   oid[NWDP_OID_MAX_LENGTH];
} NWDPOid, N_FAR *pNWDPOid;
```

Header File

nwdp_att.h

Fields

asn1Type

Specifies an Abstract Syntax Notation One (ASN1-ISO8824:1990) type byte for an object identifier. *asn1Type* should always be six (0x06).

oidLength

Specifies the number of bytes (1-127 [0x01-0x7F]).

oid

Specifies an byte array whose actual length is given by *oidLength* (worst case size is 127 [0x7F]). The array contains an object identifier encoded using the Basic Encoding Rules (BER-ISO 8825:1990).

NWDPOidCardinalMap

Contains a cardinal and optionally, an OID to mean the same thing as the cardinal

Structure

```
typedef struct {
   NWDPObjectIdentifier   oidOption;
   nuint32                cardinal;
} NWDPOidCardinalMap, N_FAR *pNWDPOidCardinalMap;
```

Header File

nwdp_att.h

Fields

oidOption

Specifies the OID synonym for the *cardinal*. (Optional)

cardinal

Specifies the number.

Remarks

NWDPOidCardinalMap corresponds to ISO 10175-1, section 9.1.5.52.

NWDPOidValueListItem

Contains one of the valid values an attribute can posses when the attribute's syntax is NWDP_AVT_ENUMERATION

Structure

```
typedef struct {
   nint32                enumValue;
   pnuint8               oidInterpretBufferPtr;
   nuint                 sizeOfResult;
   NWDPInterpretFormat   formatId;
} NWDPOidValueListItem, N_FAR *pNWDPOidValueListItem;
```

Header File

nwdp_oid.h

Fields

enumValue

Specifies the number contained in the attribute.

oidInterpretBufferPtr

Points to a temporary buffer which contains the textual interpretation of the *enumValue*.

sizeOfResult

The number of valid bytes in the *oidInterpretPtr* buffer.

formatId

Specifies the format of the result:

0 NWDP_ITPF_STRING
1 NWDP_ITPF_NUINT32_AND_STRING

Remarks

This structure allows a list of known possible values for an attribute to be presented in a callback one-by-one. The *oidInterpretBufferPtr* points to either a null-terminated string (*formatId* == 0) or to a 32-bit integer and a null-terminated string. (*formatId* ==1).

NWDPOtherFinishingSpec

Contains non-DPA finishing specifications allowed for in the syntax

Structure

```
typedef struct {
   NWDPNameOrOid          finishingOpType;
   pNWDPNameOrOid         namedSpecOptionPtr;
   NWDPOtherParameters    parameters;
} NWDPOtherFinishingSpec, N_FAR *pNWDPOtherFinishingSpec;
```

Header File

nwdp_att.h

Fields

finishingOpType

Specifies a finishing operation type.

namedSpecOptionPtr

Points to an optional named finishing specification.

parameters

Specifies parameters that accompany the finishing name.

Remarks

NWDPOtherFinishingSpec corresponds to ISO 10175-1, section 9.13.8.15.

NWDPOtherParameters

Contains "other" parameters

Structure

```
typedef struct {
   NWDPCommonParameters   common;
   pNWDPDimension         processOffsetOptionPtr;
   pNWDPLocations         headLocationsOptionPtr;
   struct {
      nuint    itemCount;
      pnint8   itemPtr;
   } otherParams;
} NWDPOtherParameters, N_FAR *pNWDPOtherParameters;
```

Header File

nwdp_att.h

Fields

common

Specifies generic finishing process parameters; corresponds to ISO 10175-1, section 9.13.8.15.4.

processOffsetOptionPtr

Points to an offset from the reference edge; corresponds to ISO 10175-1, section 9.13.8.3.6. (Optional)

headLocationsOptionPtr

Points to an head location set; corresponds to ISO 10175-1: section 9.13.8.3.7. (Optional)

itemCount

Specifies the number of array elements or items.

itemPtr

Points to the first element in the array.

Remarks

NWDPOtherParameters corresponds to ISO 10175-1 section 9.13.8.15 and its subordinate sections.

NWDPOutBinsCharacteristics

Contains a variable length array of NWDPOutputBinChar structures

Structure

```
typedef struct {
   nuint                 itemCount;
   pNWDPOutputBinChar    itemPtr;
} NWDPOutBinsCharacteristics, N_FAR *pNWDPOutBinsCharacteristics;
```

Header File

nwdp_att.h

Fields

itemCount

Specifies the number of elements in the array.

itemPtr

Points to the first element in the contiguous array.

Remarks

See ISO 10175-1 section 9.4.37.

To access each element in order, use the following syntax:

```
nint index;
NWDPOutBinsCharacteristics array;
for (index = 0; index < array.itemCount; index++)
{
if (array.itemPtr[index].stackingOrder == 1)
break;
}
if (index == array.itemCount)
printf("Not found.\n");
```

NWDPOutputBinChar

Contains a description of the characteristics of the printer's output bin

Structure

```
typedef struct {
   NWDPPageOrderTypeEnum     stackingOrder;
   NWDPPageOrientationEnum   orientation;
} NWDPOutputBinChar, N_FAR *pNWDPOutputBinChar;
```

Header File

nwdp_att.h

Fields

stackingOrder

Specifies the type of stacking order:

0 NWDP_PAGE_ORDER_UNKNOWN
1 NWDP_PAGE_ORDER_FIRST_TO_LAST
2 NWDP_PAGE_ORDER_LAST_TO_FIRST

orientation

Specifies the type of orientation:

0 NWDP_PAGE_FACE_UNKNOWN
1 NWDP_PAGE_FACE_UP
2 NWDP_PAGE_FACE_DOWN

Remarks

NWDPOutputBinChar corresponds to ISO 10175-1 section 9.4.37.

NWDPPageIdentifier

Contains information about the page numbering of a document

Structure

```
typedef struct {
   NWDPPageIdentifierEnum   designator;
   union {
      nint32          nominalPageNumber;
      NWDPText        alphanumericPageNumber;
      NWDPNameOrOid   pageTag;
   } u;
} NWDPPageIdentifier, N_FAR *pNWDPPageIdentifier;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of page identifier:

0 NWDP_PAGE_ID_NOMINAL_NUMBER
1 NWDP_PAGE_ID_ALPHANUMERIC
2 NWDP_PAGE_ID_TAG

nominalPageNumber

Specifies the in-sequence number within the document in the order that the pages are presented to the server, starting with one.

alphanumericPageNumber

Specifies the page number that can be recognized in the page identifiers of the document format.

pageTag

Specifies a group of pages.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPPageIdentifier corresponds to ISO 10175-1 section 9.3.2.20.

NWDPPageIdentifierOption

Contains an optional Page Identifier

Structure

```
typedef struct {
   nint32               flag;
   NWDPPageIdentifier   value;
} NWDPPageIdentifierOption, N_FAR *pNWDPPageIdentifierOption;
```

Header File

nwdp_att.h

Fields

flag

Specifies a Boolean value; TRUE means the *value* field is valid.

value

Specifies a page identifier if the *flag* field is non-zero.

Remarks

NWDPPageIdentifierOption corresponds to ISO 10175-1 section 9.3.2.20.

NWDPPageMediaSelect

Contains specifications regarding certain pages to be printed on specific media

Structure

```
typedef struct {
   NWDPMediaSelectEnum   designator;
   union {
      NWDPNameOrOid        mediumForAllPages;
      NWDPMediaSelectSeq   mediumForSelectedPagesSeq;
   } u;
} NWDPPageMediaSelect, N_FAR *pNWDPPageMediaSelect;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of media selected:

0 NWDP_MEDIA_SELECT_ALL_PAGES
1 NWDP_MEDIA_SELECT_SELECTED_PAGES

mediumForAllPages

Specifies simple case medium for all pages in the document.

mediumForSelectedPageSeq

Specifies a set of medium-and-page-range pairings for the document.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPPageMediaSelect corresponds to ISO 10175-1 section 9.3.2.21.

NWDPPageSelect

Contains a range of pages

Structure

```
typedef struct {
   NWDPPageIdentifierOption   beginningPage;
   NWDPPageIdentifierOption   endingPage;
} NWDPPageSelect, N_FAR *pNWDPPageSelect;
```

Header File

nwdp_att.h

Fields

beginningPage

Specifies the beginning page of the range If absent then the first page is assumed.

endingPage

Specifies the ending page of the range If absent then the last page is assumed.

Remarks

NWDPPageSelect corresponds to ISO 10175-1 section 9.3.2.20.

NWDPPageSelectSequence

Contains a variable length array of NWDPPageSelect structures

Structure

```
typedef struct {
   nuint              itemCount;
   pNWDPPageSelect    itemPtr;
} NWDPPageSelectSequence, N_FAR *pNWDPPageSelectSequence;
```

Header File

nwdp_att.h

Fields

itemCount

Specifies the number of elements in the array.

itemPtr

Points to the first element in the contiguous array.

Remarks

NWDPPageSelectSequence corresponds to ISO 10175-1 section 9.3.2.20.

To access each element in order, use the following syntax:

```
nint index;
NWDPPageSelect array;
for (index = 0; index < array.itemCount; index++)
{
   if (array.itemPtr[index].beginningPage == 0)
       break;
}
if (index == array.itemCount)
   printf("Not found.\n");
```

NWDPPageSize

Contains either the name of the page size or the dimensions of the medium

Structure

```
typedef struct {
   NWDPPageSizeEnum   designator;
   union {
      NWDPObjectIdentifier   pageSizeId;
      NWDPXYRealDimensions   pageDimensions;
   } u;
} NWDPPageSize, N_FAR *pNWDPPageSize;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of page size.

0 NWDP_PAGE_SIZE_ID
1 NWDP_PAGE_SIZE_DIMENSIONS

pageSizeId

Specifies the medium; for example, "A4 Envelope."

pageDimensions

Specifies the dimensions of the medium.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPPageSize corresponds to ISO 10175-1 section 9.4.16.

NWDPPerforatingSpec

Contains information about a finishing process which applies one or more runs of perforations perpendicular to the reference edge

Structure

```
typedef struct {
   NWDPPerforatingEnum   designator;
   union {
      NWDPNameOrOid        namedPerforating;
      NWDPPerfParameters   parameters;
   } u;
} NWDPPerforatingSpec, N_FAR *pNWDPPerforatingSpec;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of perforating specification:

0 NWDP_PERFORATING_NAMED
1 NWDP_PERFORATING_PARAMETERS

namedPerforating

Specifies a perforation operation; corresponds with ISO 10175-1: section 9.13.8.10.1.

parameters

Specifies a set of perforation parameters; corresponds with ISO 10175-1: section 9.13.8.10.2.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPPerforatingSpec corresponds with ISO 10175-1: section 9.13.8.10.

NWDPPerfParameters

Contains information about the perforation operation to perform as part of a finishing process

Structure

```
typedef struct {
   NWDPCommonParameters   common;
   NWDPLocations          e;
   NWDPNameOrOid          perfType;
} NWDPPerfParameters, N_FAR *pNWDPPerfParameters;
```

Header File

nwdp_att.h

Fields

common

Specifies generic finishing process parameters; corresponds with ISO 10175-1: section 9.13.8.10.2.

headLocations

Specifies a head location set; corresponds to ISO 10175-1: section 9.13.8.3.7.

perfType

Specifies a particular type of perforating operation.

Remarks

NWDPPerfParameters corresponds with ISO 10175-1: section 9.13.8.10.2.

NWDPPositiveIntegerOrOid

Contains either a positive integer or an OID which can be interpreted as an integer

Structure

```
typedef struct {
   NWDPPositiveIntegerOrOidEnum   designator;
   union {
      NWDPObjectIdentifier   oid;
      nint32                 integer;
   } u;
} NWDPPositiveIntegerOrOid, N_FAR *pNWDPPositiveIntegerOrOid;
```

Header File

nwdp_att.h

Fields

designator

Specifies

0 NWDP_INT_OR_OID_ID
1 NWDP_INT_OR_OID_NUMBER

oid

Specifies an OID.

integer

Specifies an integer.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPPositiveIntegerOrOid corresponds to ISO 10175-1 section 9.1.5.54.

NWDPPrinterError

Contains a Printer Error from the Printer Agent or PSM

Structure

```
typedef struct {
   NWDPPrinterProblem         problem;
   NWDPObjectIdentification   objectIdentification;
   pNWDPNameOrOid             messageOptionPtr;
} NWDPPrinterError, N_FAR *pNWDPPrinterError;
```

Header File

nwdp_att.h

Fields

problem

Specifies the nature of the error.

objectIdentification

Specifies the printer or component with the problem.

messageOptionPtr

Points to a message returned from the printer. (Optional)

Remarks

NWDPPrinterError corresponds to ISO 10175-1 section 8.4.4.

NWDPPrinterProblem

Contains the description of the printer problem

Structure

```
typedef struct {
   NWDPProblemTypeEnum   designator;
   union {
      NWDPPrinterProblemEnum   standardProblem;
      NWDPObjectIdentifier     extendedProblem
   } u;
} NWDPPrinterProblem, N_FAR *pNWDPPrinterProblem;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of problem:

0 NWDP_PROBLEM_TYPE_STANDARD
1 NWDP_PROBLEM_TYPE_EXTENDED

standardProblem

Specifies the type of standard problem:

0 NWDP_PRINTER_ERROR
1 NWDP_PRINTER_NEEDS_ATTENTION
2 NWDP_PRINTER_NEEDS_KEY_OPERATOR

extendedProblem

Specifies an OID describing the problem.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPPrinterProblem corresponds to ISO 10175-1 section 8.4.4.

NWDPPrinterStateReason

Contains information related to the condition which caused the printer state

Structure

```
typedef struct {
   NWDPNameOrOid              identifier;
   NWDPStateSeverityEnum      severity;
   NWDPTrainingEnum           trainingLevel;
   NWDPObjectIdentifier       objectClassOid;
   NWDPObjectIdentification   objectIdentification;
   nuint32                    time;
   pNWDPNameOrOid             messageOptionPtr;
} NWDPPrinterStateReason, N_FAR *pNWDPPrinterStateReason;
```

Header File

nwdp_att.h

Fields

identifier

Identifies the problem with the printer.

severity

Specifies the severity level:

1 NWDP_STATE_SEVERITY_OTHER
2 NWDP_STATE_SEVERITY_WARNING
3 NWDP_STATE_SEVERITY_CRITICAL

trainingLevel

Specifies the training level:

1 NWDP_TRAINING_OTHER
2 NWDP_TRAINING_UNKNOWN
3 NWDP_TRAINING_UNTRAINED
4 NWDP_TRAINING_TRAINED
6 NWDP_TRAINING_FIELD_SERVICE
6 NWDP_TRAINING_MANAGEMENT

objectClassOid

Specifies the class of the object which has the problem.

objectIdentification

Specifies the object within the class which has the problem.

time

Specifies the time when the problem was reported.

messageOptionPtr

Points to a message which describes the problem. (Optional)

Remarks

The NWDPPrinterStateReason structure contains only problems such as "paper out" or "toner low."

NWDPProfileResultSet

Contains a variable length array of NWDPEventHandlingProfile structures

Structure

```
typedef struct {
   nuint                        itemCount;
   pNWDPEventHandlingProfile    itemPtr;
} NWDPProfileResultSet, N_FAR *pNWDPProfileResultSet;
```

Header File

nwdp_att.h

Fields

itemCount

Specifies the number of elements in the array.

itemPtr

Points to the first element in the contiguous array.

Remarks

To access each element in order, use the following syntax:

```
nint index;
NWDPEventHandlingProfile array;
for (index = 0; index < array.itemCount; index++)
{
   if (array.itemPtr[index].profileId == 0)
       break;
}
if (index == array.itemCount)
   printf("Not found.\n");
```

NWDPPrtConfigObjectId

Contains the identification of a Job or Document configuration object

Structure

```
typedef struct {
   NWDPText            printerName;
   NWDPQualifiedName   qualifiedName;
} NWDPPrtConfigObjectId, N_FAR *pNWDPPrtConfigObjectId;
```

Header File

nwdp_att.h

Fields

printerName

Specifies the name of the printer agent.

qualifiedName

Specifies the printer object associated with the printer agent.

Remarks

The NWDPPrtConfigObjectId structure contains the identifier for one of the following classes:

```
Doc_Defaults
Doc_Limits
Job_Defaults
Job_Limits
```

NWDPPrtContainedObjectId

Contains the identifier of an object contained by a printer agent

Structure

```
typedef struct {
   NWDPText   printerName;
   nuint32    localIdentifier;
} NWDPPrtContainedObjectId, N_FAR *pNWDPPrtContainedObjectId;
```

Header File

nwdp_att.h

Fields

printerName

Specifies the name of the printer agent which contains the specified object.

localIdentifier

Specifies the identifier of the object which is contained by the printer agent.

Remarks

The NWDPPrtContainedObjectId structure contains the identifier for one of the following classes:

```
Buffer
Channel
Cover
Input
Interpreter
Job
```
Light
```
Marker
Marker_Colorant
Marker_Supplies
Media_Path
Output
Retained_job
```

NWDPPrtContainedObjectIdSet

Contains a variable length array of NWDPPrtContainedObjectId structures

Structure

```
typedef struct {
   nuint                          itemCount;
   pNWDPPrtContainedObjectId      itemPtr;
} NWDPPrtContainedObjectIdSet, N_FAR *pNWDPPrtContainedObjectIdSet;
```

Header File

nwdp_att.h

Fields

itemCount

Specifies the number of elements in the array.

itemPtr

Points to the first element in the contiguous array.

Remarks

To access each element in order, use the following syntax:

```
nint index;
NWDPPrtContainedObjectId array;
for (index = 0; index < array.itemCount; index++)
{
   if (array.itemPtr[index].printerName == 0)
       break;
}
if (index == array.itemCount)
   printf("Not found.\n");
```

NWDPPrtGlobalListItem

Describes a specified printer within the name service

Structure

```
typedef struct
{
   nchar16              printerName16[NWDP_PRT_MAX_NAME_CHARS];
   nchar16              description16[NWDP_PRT_MAX_DESCRIPTION_CHARS];
   nchar16              location16[NWDP_PRT_MAX_LOCATION_CHARS];
   NWDPPrtRefTypeEnum   printerType;
   union {
      struct {
         NWDPNetAddress   address;
      } convenience;
      NWDPNSrvFQNBuffer   named;
   } u;
} NWDPPrtGlobalListItem, N_FAR *pNWDPPrtGlobalListItem;
```

Header File

nwdp_prt.h

Fields

printerName16

Specifies the manufacturers name for the printer.

description16

Specifies the locality attribute from the name service.

location16

Specifies the attribute description from the name service.

printerType

Specifies the type of printer.

NWDPPrtInstalledListItem

Describes a specified printer from within the installed list on the workstation

Structure

```
typedef struct
{
   nbool     defaultPrinter;
   nchar16   name16[NWDP_PRT_MAX_NAME_CHARS+1];
   nchar16   driver16[NWDP_PRT_MAX_DRIVER_CHARS+1];
   nchar16   label16[NWDP_PRT_MAX_LABEL_CHARS+1];
   nchar16   port16[NWDP_PRT_MAX_PORT_CHARS+1];
} NWDPPrtInstalledListItem, N_FAR *pNWDPPrtInstalledListItem;
```

Header File

nwdp_prt.h

Fields

defaultPrinter

Specifies if the printer is the default:

N_TRUE Is the default printer
N_FALSE Is not the default printer

name16

Specifies the manufacturing name for the printer.

driver16

Specifies the name of the printer driver (required).

label16

Specifies the name used when calling **NWDPPrtCreateRefBasedOnLabel**.

port16

Specifies the port to which the installed printer will print.

NWDPPrtJobListItem

Describes a submitted job

Structure

```
typedef struct
{
   nuint32    jobID;
   nuint32    jobPosition;
   pnstr16    jobName16Ptr;
   pnstr16    jobOwner16Ptr;
   nuint32    jobSize;
   nuint      numDocs;
   nuint      jobPriority;
   nuint32    submitTime;
   pNWDPOid   jobStateOidPtr;
} NWDPPrtJobListItem, N_FAR *pNWDPPrtJobListItem;
```

Header File

nwdp_prt.h

Fields

jobID

Specifies the identifier from the virtual printer.

jobName16Ptr

Specifies the jobs relative position in the queue.

jobOwner16Ptr

Specifies the job name.

jobSize

Specifies the name of the owner (not necessarily the creator).

numDocs

Specifies the number of bytes of data in the job.

jobPriority

Specifies the number of documents in the job.

submitTime

Specifies the time the job was submitted.

jobStateOidPtr

Specifies the current state of the job presented as an OID.

NWDPPsmPAListItem

Contains information about all the printer agents on the given PSM

Structure

```
typedef struct {
   pnstr16    paName16Ptr;
   pnstr16    phStartup16Ptr;
   pnstr16    pdsStartup16Ptr;
   pNWDPOid   paStateOidPtr;
   nint32     printerAgentId;
} NWDPPsmPAListItem, N_FAR *pNWDPPsmPAListItem;
```

Header File

nwdp_psm.h

Fields

paName16Ptr

Points to the printer agent name.

phStartup16Ptr

Points to the port handler startup string, which is passed as a command line parameter when the port handler is loaded on the server.

pdsStartup16Ptr

Points to the print device subsystem (PDS) startup string, which is passed as a command line parameter when the PDS is loaded on the server.

paStateOidPtr

Points to the OID specifying the state of the printer agent. Use **NWDPOidInterpretRef** to get an interpretation for the OID.

printerAgentId

Specifies whether the printer is enabled; if N_TRUE, the printer can receive jobs.

Remarks

None

NWDPPunchingSpec

Contains information about the punching operation which is part of a finishing process

Structure

```
typedef struct {
   NWDPPunchingEnum   designator;
   union {
      NWDPNameOrOid         namedPunching;
      NWDPPunchParameters   parameters;
   } u;
} NWDPPunchingSpec, N_FAR *pNWDPPunchingSpec;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of punching specification:

0 NWDP_PUNCHING_NAMED
1 NWDP_PUNCHING_PARAMETERS

namedPunching

Specifies a particular punching operation; corresponds with ISO 10175-1: section 9.13.8.9.1.

parameters

Specifies a set of punching parameters; corresponds with ISO 10175-1: section 9.13.8.9.2.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPPunchingSpec corresponds with ISO 10175-1: section 9.13.8.9.

NWDPPunchParameters

Contains parameters that control the punching operation which is part of a finishing process

Structure

```
typedef struct {
   NWDPCommonParameters    common;
   pNWDPDimension          processOffsetOptionPtr;
   NWDPLocations           headLocations;
   pNWDPDimension          punchDiameterPtr;
} NWDPPunchParameters, N_FAR *pNWDPPunchParameters;
```

Header File

nwdp_att.h

Fields

common

Specifies generic finishing process parameters.

processOffsetOptionPtr

Points to an offset from the reference edge; corresponds to ISO 10175-1, section 9.13.8.3.6. (Optional)

headLocations

Specifies a head location set; corresponds to ISO 10175-1: section 9.13.8.3.7.

punchDiameterPtr

Points to a system-dependent value for the punch diameter.

Remarks

None

NWDPQualifiedName

Contains a qualified name

Structure

```
typedef struct {
   NWDPQualifiedNameEnum   designator;
   union {
      NWDPText       simpleName;
      NWDPNdsName    ndsName;
   } u;
} NWDPQualifiedName, N_FAR *pNWDPQualifiedName;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of qualified name:

0 NWDP_QUALIFIED_NAME_NONE
1 NWDP_QUALIFIED_NAME_NDS

simpleName

Specifies the simple (text) name.

ndsName

Specifies the distinguished NDS name.

Remarks

NDPSQualifiedName is used to contain any instance of an NDS name. It also supports the text type for such items as public access printer agents. There is a one-to-one correspondence between the designator values and the fields within the union.

NWDPQualifiedNameMap

Contains a pair of associated qualified names

Structure

```
typedef struct {
   NWDPQualifiedName   primary;
   NWDPQualifiedName   secondary;
} NWDPQualifiedNameMap, N_FAR *pNWDPQualifiedNameMap;
```

Header File

nwdp_att.h

Fields

primary

Specifies the primary qualified name.

secondary

Specifies the secondary qualified name.

Remarks

NWDPQualifiedNameMap is used to represent the association of two objects; for example, a printer agent and an associated printer object.

NWDPQualifiedNameSet

Contains a variable length array of NWDPQualifiedName structures

Structure

```
typedef struct {
   nuint                 itemCount;
   pNWDPQualifiedName    itemPtr;
} NWDPQualifiedNameSet, N_FAR *pNWDPQualifiedNameSet;
```

Header File

nwdp_att.h

Fields

itemCount

Specifies the number of elements in the array.

itemPtr

Points to the first element in the contiguous array.

Remarks

To access each element in order, use the following syntax:

```
nint index;
NWDPQualifiedName array;
for (index = 0; index < array.itemCount; index++)
{
   if (array.itemPtr[index].designator == 0)
       break;
}
if (index == array.itemCount)
   printf("Not found.\n");
```

NWDPRDNSequence

Contains an array of relative distinguished names (RDNs)

Structure

```
typedef struct {
   nuint       itemCount;
   pNWDPText   itemPtr;
} NWDPRDNSequence, N_FAR *pNWDPRDNSequence;
```

Header File

nwdp_att.h

Fields

itemCount

Specifies the number of elements in the array.

itemPtr

Points to the first element in the array.

Remarks

NWDPRDNSequence corresponds to ISO 10175-1: section 9.1.5.41.

NWDPRealOption

Contains an optional floating point value

Structure

```
typedef struct {
   nint32    flag;
   nreal64   value;
} NWDPRealOption, N_FAR *pNWDPRealOption;
```

Header File

nwdp_att.h

Fields

flag

Specifies whether *value* has meaning:

0 *value* has no meaning
non-zero *value* has meaning

value

Specifies a real (floating point) number in IEEE 64-bit format.

Remarks

None

NWDPRealRange

Contains a range of allowable values for the Real datatype

Structure

```
typedef struct {
   nreal64   lowerBound;
   nreal64   upperBound;
} NWDPRealRange, N_FAR *pNWDPRealRange;
```

Header File

nwdp_att.h

Fields

lowerBound

Specifies the lower boundary of the value range.

upperBound

Specifies the upper boundary of the value range.

Remarks

NWDPRealRange corresponds to ISO 10175-1: section 9.1.5.32.

NWDPRealSet

Contains a variable length array of Real values

Structure

```
typedef struct {
   nuint      itemCount;
   pnreal64   itemPtr;
} NWDPRealSet, N_FAR *pNWDPRealSet;
```

Header File

nwdp_att.h

Fields

itemCount

Specifies the number of elements in the array.

itemPtr

Points to the first element in the contiguous array.

Remarks

NWDPRealSet corresponds with ISO 10175-1: section 9.1.5.30.

NWDPResAddFontFile

Contains font file information

Structure

```
typedef struct {
   NWDPOSTypeEnum     osType;
   NWDPFontTypeEnum   fontType;
   pnstr16            fontName16Ptr;
   pnstr16            fileName16Ptr;
} NWDPResAddFontFile, N_FAR *pNWDPResAddFontFile;
```

Header File

nwdp_res.h

Fields

osType

Specifies the operating system type.

0 NWDP_RES_OS_DOS
1 NWDP_RES_OS_WIN_31
2 NWDP_RES_OS_WIN_95
3 NWDP_RES_OS_WIN_NT
4 NWDP_RES_OS_OS_2
5 NWDP_RES_OS_MAC
6 NWDP_RES_OS_UNIX

fontType

Specifies the font type.

0 NWDP_RES_FONT_TRUE_TYPE
1 NWDP_RES_FONT_PS
2 NWDP_RES_FONT_SYSTEM
3 NWDP_RES_FONT_SPD
4 NWDP_RES_FONT_TRUE_DOC

fontName16Ptr

Points to a Unicode string containing a font name (for example, Arial).

fileName16Ptr

Points to a Unicode string containing a filename (for example, arial.ttf).

Remarks

Print Service Group

Used by the **NWDPResAddResourceFile** function.

NWDPResAddPrnDefFile

Contains printer definition (.npd) file information

Structure

```
typedef struct {
   pnstr16   vendorDir16Ptr;
   psntr16   fileName16Ptr;
} NWDPResAddPrnDefFile, N_FAR *pNWDPResAddPrnDefFile;
```

Header File

nwdp_res.h

Fields

vendorDir16Ptr

Points to a Unicode string containing the name of the vendor directory (for example, Novell) in the Resource Manager data storage area.

fileName16Ptr

Points to a Unicode string containing a printer definition filename (for example, HP04.NPD).

Remarks

Used by the **NWDPResAddResourceFile** function.

NWDPResAddPrnDrvFile

Contains printer driver file information

Structure

```
typedef struct {
   NWDPOSTypeEnum   osType;
   pnstr16          driverDirName16Ptr;
   pnstr16          fileName16Ptr;
} NWDPResAddPrnDrvFile, N_FAR *pNWDPResAddPrnDrvFile;
```

Header File

nwdp_res.h

Fields

osType

Specifies the operating system type:

0 NWDP_RES_OS_DOS
1 NWDP_RES_OS_WIN_31
2 NWDP_RES_OS_WIN_95
3 NWDP_RES_OS_WIN_NT
4 NWDP_RES_OS_OS_2
5 NWDP_RES_OS_MAC
6 NWDP_RES_OS_UNIX

driverDirName16Ptr

Is set to NULL if you are adding a driver index (INF) file, or points to the Unicode string containing the characters "new" if any other driver file type is being added.

fileName16Ptr

Points to a Unicode string containing a printer driver filename.

Remarks

Used by the **NWDPResAddResourceFile** function.

The Unicode string pointed to by the *driverDirName16Ptr* field is returned in the NWDPResCallbackItem structure when the **NWDPResListResource** function is called.

NWDPResDeleteFontFile

Contains font file information

Structure

```
typedef struct {
   NWDPOSTypeEnum     osType;
   NWDPFontTypeEnum   fontType;
   pnstr16            fontName16Ptr;
   pnstr16            fileName16Ptr;
} NWDPResDeleteFontFile, N_FAR *pNWDPResDeleteFontFile;
```

Header File

nwdp_res.h

Fields

osType

Specifies the operating system type:

0 NWDP_RES_OS_DOS
1 NWDP_RES_OS_WIN_31
2 NWDP_RES_OS_WIN_95
3 NWDP_RES_OS_WIN_NT
4 NWDP_RES_OS_OS_2
5 NWDP_RES_OS_MAC
6 NWDP_RES_OS_UNIX

fontType

Specifies the font type:

0 NWDP_RES_FONT_TRUE_TYPE
1 NWDP_RES_FONT_PS
2 NWDP_RES_FONT_SYSTEM
3 NWDP_RES_FONT_SPD
4 NWDP_RES_FONT_TRUE_DOC

fontName16Ptr

Points to a Unicode string containing a font name (for example, Arial).

fileName16Ptr

Points to a Unicode string containing a font filename (for example, arial.ttf).

Remarks

Print Service Group

Used by the **NWDPResDeleteResourceFile** function.

NWDPResDeletePrnDefFile

Contains printer definition (.npd) file information

Structure

```
typedef struct {
   pnstr16   vendorDir16Ptr;
   pnstr16   fileName16Ptr;
} NWDPResDeletePrnDefFile, N_FAR *pNWDPResDeletePrnDefFile;
```

Header File

nwdp_res.h

Fields

vendorDir16Ptr

Points to a Unicode string containing the name of the vendor directory (for example, Novell) in the Resource Manager data storage area.

fileName16Ptr

Points to a Unicode string containing the printer definition filename (for example, HP04.NPD.

Remarks

Used by the **NWDPResDeleteResourceFile** function.

NWDPResDeletePrnDrvFile

Contains printer driver file information

Structure

```
typedef struct {
   NWDPOSTypeEnum   osType;
   pnstr16          driverDirName16Ptr;
   pnstr16          fileName16Ptr;
} NWDPResDeletePrnDrvFile, N_FAR *pNWDPResDeletePrnDrvFile;
```

Header File

nwdp_res.h

Fields

osType

Specifies the operating system type:

0 NWDP_RES_OS_DOS
1 NWDP_RES_OS_WIN_31
2 NWDP_RES_OS_WIN_95
3 NWDP_RES_OS_WIN_NT
4 NWDP_RES_OS_OS_2
5 NWDP_RES_OS_MAC
6 NWDP_RES_OS_UNIX

driverDirName16Ptr

Points to a Unicode string containing the name of the driver directory in the Resource Manager data storage area. This string is returned in the NWDPResCallBackItem structure when a call to the **NWDPResListResource** function is made with the NWDPResListTypeEnum set to NWDP_RES_LIST_PRN_DRV_FILES.

fileName16Ptr

Points to a Unicode string containing the printer driver filename.

Remarks

Used by the **NWDPResDeleteResourceFile** function.

NWDPResGetDatePrnDefFile

Contains printer definition (.npd) file information

Structure

```
typedef struct {
   pnstr16   vendorDir16Ptr;
   pnstr16   fileName16Ptr;
} NWDPResGetDatePrnDefFile, N_FAR *pNWDPResGetDatePrnDefFile;
```

Header File

nwdp_res.h

Fields

vendorDir16Ptr

Points to a Unicode string containing the name of the vendor directory (for example, Novell) in the Resource Manager data storage area.

fileName16Ptr

Points to a Unicode string containing a printer definition filename (for example, HP04.NPD.

Remarks

Used by the **NWDPResGetResourceFile** function.

NWDPResGetDatePrnDrvFile

Contains printer driver file information

Structure

```
typedef struct {
   NWDPOSTypeEnum   osType;
   pnstr16          driverDirName16Ptr;
   pnstr16          fileName16Ptr;
} NWDPResGetDatePrnDrvFile, N_FAR *pNWDPResGetDatePrnDrvFile;
```

Header File

nwdp_res.h

Fields

osType

Specifies the operating system type:

0 NWDP_RES_OS_DOS
1 NWDP_RES_OS_WIN_31
2 NWDP_RES_OS_WIN_95
3 NWDP_RES_OS_WIN_NT
4 NWDP_RES_OS_OS_2
5 NWDP_RES_OS_MAC
6 NWDP_RES_OS_UNIX

driverDirName16Ptr

Points to a Unicode string containing the name of the driver directory in the Resource Manager data storage area. This string is returned in the NWDPResListCallBackItem structure when a call to the **NWDPResListResource** function is made with the NWDPResListTypeEnum set to NWDP_RES_LIST_PRN_DRV_FILES.

fileName16Ptr

Points to a Unicode string containing a printer driver filename

Remarks

Used by the **NWDPResGetResourceFile** function.

NWDPResGetFontFile

Contains font file information

Structure

```
typedef struct {
   NWDPOSTypeEnum     osType;
   NWDPFontTypeEnum   fontType;
   pnstr16            fileName16Ptr;
} NWDPResGetFontFile, N_FAR *pNWDPResGetFontFile;
```

Header File

nwdp_res.h

Fields

osType

Specifies the operating system type:

0 NWDP_RES_OS_DOS
1 NWDP_RES_OS_WIN_31
2 NWDP_RES_OS_WIN_95
3 NWDP_RES_OS_WIN_NT
4 NWDP_RES_OS_OS_2
5 NWDP_RES_OS_MAC
6 NWDP_RES_OS_UNIX

fontType

Specifies the font type:

0 NWDP_RES_FONT_TRUE_TYPE
1 NWDP_RES_FONT_PS
2 NWDP_RES_FONT_SYSTEM
3 NWDP_RES_FONT_SPD
4 NWDP_RES_FONT_TRUE_DOC

fileName16Ptr

Points to a Unicode string containing a font filename (for example, arial.ttf).

Remarks

Used by the **NWDPResGetResourceFile** function.

NWDPResGetPrnDefFile

Contains printer definition (.npd) file information

Structure

```
typedef struct {
   pnstr16   vendorDir16Ptr;
   pnstr16   fileName16Ptr;
} NWDPResGetPrnDefFile, N_FAR *pNWDPResGetPrnDefFile;
```

Header File

nwdp_res.h

Fields

vendorDir16Ptr

Points to a Unicode string containing the name of the vendor directory (for example, Novell) in the Resource Manager data storage area.

fileName16Ptr

Points to a Unicode string containing a printer definition filename (for example, HP04.NPD.

Remarks

Used by the **NWDPResGetResourceFile** function.

NWDPResGetPrnDrvFile

Contains a file and a directory destination for driver downloads

Structure

```
typedef struct {
   pnstr16   driverDirName16Ptr;
   pnstr16   fileName16Ptr;
} NWDPResGetPrnDrvFile, N_FAR *pNWDPResGetPrnDrvFile;
```

Header File

nwdp_res.h

Fields

driverDirName16Ptr

Points to a Unicode string containing the name of the driver directory in the Resource Manager data storage area. This string is returned in the NWDPResListCallBackItem structure when a call is made to the **NWDPResListResource** function is made with the NWDPResListTypeEnum set to NWDP_RES_LIST_PRN_DRV_FILES.

fileName16Ptr

Points to a Unicode string containing a printer driver filename.

Remarks

Used by the **NWDPResGetResourceFile** function.

NWDPResListCallbackItem

Returns the requested resource information

Structure

```
typedef struct
{
   NWDPResListTypeEnum   listType;
   union
   {
      struct
      {
         pnstr16   printerType16Ptr;
         pnstr16   printerMfr16Ptr;
         pnstr16   infFileName16Ptr;
      } ListPrnDrvTypesCB;

      struct
      {
         pnstr16   printerType16Ptr;
         pnstr16   printerMfr16Ptr;
         pnstr16   infFileName16Ptr;
      } ListPrnDefTypesCB;

      pnstr16   bannerFileName16Ptr;
      pnstr16   fontName16Ptr;

      struct
      {
         pnstr16   fileName16Ptr;
         pnstr16   driverDirName16Ptr;
         pnstr16   infFileName16Ptr;
      } ListPrnDrvFilesCB;

      struct
      {
         pnstr16             defFileNamePtr;
         NWDPResUnicodeSet   win31KeySet;
         NWDPResUnicodeSet   win95KeySet;
      } ListPrnDefFilesCB;

      pnstr16   fontFileName16Ptr;
   } callback_u;
} NWDPResListCallbackItem, N_FAR *pNWDPResListCallbackItem;
```

Header File

nwdp_res.h

Fields

listType

Specifies the list type enumeration values passed to **NWDPResListResource**.

printerType16Ptr

Points to a Unicode string containing the "readable" printer type.

printerMfr16Ptr

Points to a Unicode string containing the printer manufacturer name.

infFileName16Ptr

Points to a Unicode string containing the name of the index file printer type.

printerType16Ptr

Points to a Unicode string containing the "readable" printer type.

printerMfr16Ptr

Points to a Unicode string containing the printer manufacturer name.

infFileName16Ptr

Points to a Unicode string containing the name of the index file printer type.

bannerFileName16Ptr

Points to a Unicode string containing the banner file name.

fontName16Ptr

Points to a Unicode string containing the "readable" font name.

fileName16Ptr

Points to a Unicode string containing the filename of the driver component file.

driverDirName16Ptr

Points to a Unicode string containing the name of the resource manager subdirectory where component driver files are found (needed to call **NWDPResGetResourceFile**, **NWDPResAddResourceFile**, and **NWDPResDeleteResourceFile** functions.

defFileName16Ptr

Points to a Unicode string containing the printer definition file name (.NPD filename).

win31KeySet

NWDPResUnicodeSet containing (in Unicode) matching printer type name(s) that match those in a Windows 3.1 printer .INF file.

win95KeySet

NWDPResUnicodeSet containing (in Unicode) matching printer type name(s) that match those in a Windows95 printer .INF file.

fontFileName16Ptr

Points to a Unicode string containing the name of the actual font file that will be used to call the **NWDPResGetResourceFile** function.

Remarks

Used by the **NWDPResListResource** function.

NWDPResListDefFiles

Contains information needed to obtain a list of printer definition files

Structure

```
typedef struct {
   pnstr16            vendorDir16Ptr;
   pnstr16            printerType16Ptr;
   pnstr16            printerMfr16Ptr;
   pnstr16            infFileName16Ptr;
   pNWDPOctetString   printerIdPtr;
} NWDPResListDefFiles, N_FAR *pNWDPResListDefFiles;
```

Header File

nwdp_res.h

Fields

vendorDir16Ptr

Points to a Unicode string containing the name of the vendor directory in the Resource Manager data storage area.

printerType16Ptr

Points to a Unicode string containing the "readable" printer type.

printerMfr16Ptr

Points to a Unicode string containing the printer manufacturer name.

infFileName16Ptr

Points to a Unicode string containing the index (.INF) filename and path.

printerIdPtr

Points to a Unicode string containing the P1284 Device ID string from a printer.

Remarks

Input for this structure comes from data returned in the NWDPResCallbackItem structure after the **NWDPResListResource** function is called with a *resListType* parameter of NWDP_RES_LIST_PRN_DEF_TYPES.

The contents of this structure change based on the information available as follows:

If the *printerIdPtr* field (P1284 DID) is not available, the structure fields

vendorDir16Ptr, *printerType16Ptr*, and *infFileName16Ptr* must have valid values and *printerIdPtr* must be set to NULL.

If the *printerIdPtr* field (P1284 DID) is available, the first four fields of this structure must be set to NULL.

Used by the **NWDPResListResource** function.

NWDPResListDrvFiles

Contains information needed to obtain a list of printer driver files

Structure

```
typedef struct {
   NWDPOSTypeEnum     osType;
   pnstr16            printerType16Ptr;
   pnstr16            printerMfr16Ptr;
   pnstr16            infFileName16Ptr;
   pNWDPOctetString   printerIdPtr;
} NWDPResListDrvFiles, N_FAR *pNWDPResListDrvFiles;
```

Header File

nwdp_res.h

Fields

osType

Specifies the operating system type:

0 NWDP_RES_OS_DOS
1 NWDP_RES_OS_WIN_31
2 NWDP_RES_OS_WIN_95
3 NWDP_RES_OS_WIN_NT
4 NWDP_RES_OS_OS_2
5 NWDP_RES_OS_MAC
6 NWDP_RES_OS_UNIX

printerType16Ptr

Points to a Unicode string containing the "readable" printer type.

printerMfr16Ptr

Points to a Unicode string containing the printer manufacturer name.

infFileName16Ptr

Points to a Unicode string containing the index (.INF) filename and path.

printerIdPtr

Points to a Unicode string containing the P1284 Device ID string from a printer.

Remarks

Input for this structure comes from data returned in the

NWDPResCallbackItem structure after the **NWDPResListResource** function is called with a *resListType* parameter of NWDP_RES_LIST_PRN_DRV_TYPES.

The contents of this structure change based on the information available as follows:

If the *printerIdPtr* field (P1284 DID) is not available, the structure fields *vendorDir16Ptr*, *printerType16Ptr*, and *infFileName16Ptr* must have valid values and *printerIdPtr* must be set to NULL.

If the *printerIdPtr* field (P1284 DID) is available, the first four fields of this structure must be set to NULL.

Used by the **NWDPResListResource** function.

NWDPResListFontFiles

Contains information needed to obtain a list of font files

Structure

```
typedef struct {
   NWDPOSTypeEnum     osType;
   NWDPFontTypeEnum   fontType;
   pnstr16            fontName16Ptr;
} NWDPResListFontFiles, N_FAR *pNWDPResListFontFiles;
```

Header File

nwdp_res.h

Fields

osType

Specifies the operating system type:

0 NWDP_RES_OS_DOS
1 NWDP_RES_OS_WIN_31
2 NWDP_RES_OS_WIN_95
3 NWDP_RES_OS_WIN_NT
4 NWDP_RES_OS_OS_2
5 NWDP_RES_OS_MAC
6 NWDP_RES_OS_UNIX

fontType

Specifies the font type:

0 NWDP_RES_FONT_TRUE_TYPE
1 NWDP_RES_FONT_PS
2 NWDP_RES_FONT_SYSTEM
3 NWDP_RES_FONT_SPD
4 NWDP_RES_FONT_TRUE_DOC

fontName16Ptr

Points to a Unicode string containing a font name (for example, Arial).

Remarks

Used by the **NWDPResListResource** function.

NWDPResListFontTypes

Contains information needed to obtain a list of available font types

Structure

```
typedef struct {
   NWDPFontTypeEnum   fontType;
   NWDPOSTypeEnum     osType;
} NWDPResListFontTypes, N_FAR *pNWDPResListFontTypes;
```

Header File

nwdp_res.h

Fields

fontType

Specifies the font type:

0 NWDP_RES_FONT_TRUE_TYPE
1 NWDP_RES_FONT_PS
2 NWDP_RES_FONT_SYSTEM
3 NWDP_RES_FONT_SPD
4 NWDP_RES_FONT_TRUE_DOC

osType

Specifies the operating system type:

0 NWDP_RES_OS_DOS
1 NWDP_RES_OS_WIN_31
2 NWDP_RES_OS_WIN_95
3 NWDP_RES_OS_WIN_NT
4 NWDP_RES_OS_OS_2
5 NWDP_RES_OS_MAC
6 NWDP_RES_OS_UNIX

Remarks

None

NWDPResource

Contains the identifier of an object used to render the printed output

Structure

```
typedef struct {
   NWDPResourceEnum   designator;
   union {
      NWDPNameOrOid   nameOrOid;
      NWDPText        name;
   } u;
} NWDPResource, N_FAR *pNWDPResource;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of resource:

0 NWDP_RESOURCE_NAME_OR_OID
1 NWDP_RESOURCE_TEXT

nameOrOid

Specifies the object used to render the printed output.

name

Specifies the name of the object used to render the printed output.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. See ISO10175-1: section 6.5.3.

NWDPResourceContext

Contains server-specific information to identify a resource in a way that is server independent

Structure

```
typedef struct {
   NWDPResourceContextEnum   designator;
   union {
      NWDPNameOrOid   objectName;
      NWDPText        pathName;
   } u;
} NWDPResourceContext, N_FAR *pNWDPResourceContext;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of resource context:

0 NWDP_RESOURCE_CONTXT_NAME_OR_OID
1 NWDP_RESOURCE_CONTEXT_TEXT

objectName

Specifies the name of the object.

pathName

Specifies the path to the object.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPResourceContext corresponds to ISO 10175-1 section 9.19.

NWDPResourceContextSeq

Contains a variable length array of NWDPResourceContext structures

Structure

```
typedef struct {
   nuint                   itemCount;
   pNWDPResourceContext    itemPtr;
} NWDPResourceContextSeq, N_FAR *pNWDPResourceContextSeq;
```

Header File

nwdp_att.h

Fields

itemCount

Specifies the number of elements in the array.

itemPtr

Points to the first element in the contiguous array.

Remarks

NWDPResourceContextSeq corresponds to ISO 10175-1 section 9.19.

To access each element in order, use the following syntax:

```
nint index;
NWDPResourceContext array;
for (index = 0; index < array.itemCount; index++)
{
   if (array.itemPtr[index].designator == 0)
       break;
}
if (index == array.itemCount)
   printf("Not found.\n");
```

NWDPResourcePrinterId

Contains information about the printer driver

Structure

```
typedef struct {
   NWDPText   keyName;
   NWDPText   mfrName;
   NWDPText   infFileName;
   nint32     osType;
} NWDPResourcePrinterId, N_FAR *pNWDPResourcePrinterId;
```

Header File

nwdp_att.h

Fields

keyName

Specifies the printer driver key name in Unicode.

mfrName

Specifies the printer manufacturer name in Unicode.

infFileName

Specifies the name of the .INF file in which the printer driver information for installation is found.

osType

Specifies the operating system type

Remarks

The data in the NWDPResourceID structure is part of the Managed Object Database (MOD) and is filled in when a printer object is created.

NWDPRestrictionInfo

Contains information about restrictions on delivery addresses

Structure

```
typedef struct {
   nbool    restrictedBool;
   nint32   min;
   nint32   max;
} NWDPRestrictionInfo, N_FAR *pNWDPRestrictionInfo;
```

Header File

nwdp_nto.h

Fields

restrictedBool

Specifies that a delivery address is restricted.

min

Specifies a minimum value for a delivery address item.

max

Specifies a maximum value for a delivery address item.

Remarks

NWDPRestrictionInfo is contained within the NWDPDeliveryAddrInfo structure and is the mechanism by which a delivery method specifies if there is a range of valid values for a delivery address and if there is a range, what are the limits.

NWDPResultsProfile

Contains the profile for determining how to send the printed job results

Structure

```
typedef struct {
   NWDPObjectIdentifier       deliveryMethodOption;
   pNWDPNameOrOid             resultsSetCommentOptionPtr;
   pNWDPDeliveryAddress       deliveryAddressOptionPtr;
   nint32                     jobCopies;
   NWDPPrtContainedObjectId   outputBinOption;
} NWDPResultsProfile, N_FAR *pNWDPResultsProfile;
```

Header File

nwdp_att.h

Fields

deliveryMethodOption

Specifies a delivery method. If omitted the default is used. (Optional)

resultsSetCommentOptionPtr

Specifies a human-readable string which helps in the delivery of the hard copy results of the job. (Optional)

deliveryAddressOptionPtr

Specifies a machine readable delivery address whose format is specific to the method of delivery. (Optional)

jobCopies

Specifies the number of copies to be delivered to this addressee.

outputBinOptionPtr

Specifies the output bin where the jobs will be left. (Optional)

Remarks

NWDPResultsProfile corresponds to ISO 10175-1 section 9.2.2.1.

NWDPResUnicodeSet

Contains a set of Unicode strings

Structure

```
typedef struct {
 nint     setCount;
 pnstr16  setItem16Ptr;
} NWDPResUnicodeSet, N_FAR *pNWDPResUnicodeSet;
```

Header File

nwdp_res.h

Fields

setCount

Specifies the number of array elements (Unicode strings) in the *setItem16Ptr* field.

setItem16Ptr

Points to array of pnstr16 pointers.

Remarks

NWDPResUnicodeSet is used in the NWDPResListCallbackItem callback structure.

NWDPSecurityError

Contains a description of a security error from the PSM

Structure

```
typedef struct {
   NWDPSecurityProblem   problem;
   pNWDPNameOrOid        messageOptionPtr;
} NWDPSecurityError, N_FAR *pNWDPSecurityError;
```

Header File

nwdp_att.h

Fields

problem

Specifies the security problem.

messageOptionPtr

Specifies a message from the PSM to the caller. (Optional)

Remarks

NWDPSecurityError corresponds to ISO 10175-1: section 8.4.5.

NWDPSecurityProblem

Specifies a sufficient rights problem with the request sent to the printer agent or PSM

Structure

```
typedef struct {
   NWDPProblemTypeEnum   designator;
   union {
      NWDPSecurityProblemEnum   standardProblem;
      NWDPObjectIdentifier      extendedProblem;
   } u;
} NWDPSecurityProblem, N_FAR *pNWDPSecurityProblem;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of problem:

0 NWDP_PROBLEM_TYPE_STANDARD
1 NWDP_PROBLEM_TYPE_EXTENDED

standardProblem

Specifies the type of standard problem:

0 NWDP_SECURITY_AUTHENTICATION
1 NWDP_SECURITY_CREDENTIALS
2 NWDP_SECURITY_RIGHTS
3 NWDP_SECURITY_INVALID_PAC

extendedProblem

Specifies an OID describing the extended problem.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPSecurityProblem corresponds to ISO 10175-1 section 8.4.5.

NWDPSelectionError

Contains a description of the error in a request to the Printer Agent or PSM where the specified object was not found

Structure

```
typedef struct {
   NWDPSelectionProblem       problem;
   NWDPObjectIdentification   objectIdentification;
   pNWDPAttribute             attributeOptionPtr;
   pNWDPNameOrOid             messageOptionPtr;
} NWDPSelectionError, N_FAR *pNWDPSelectionError;
```

Header File

nwdp_att.h

Fields

problem

Specifies the problem with the request.

objectIdentification

Specifies the object which could not be selected.

attributeOptionPtr

Points to the attribute of the object which could not be selected. (Optional)

messageOptionPtr

Points to a message from the PSM to the caller. (Optional)

Remarks

NWDPSelectionError corresponds to ISO 10175-1 section 8.4.7.

NWDPSelectionProblem

Contains details related to the NWDPSelectionError structure

Structure

```
typedef struct {
   NWDPProblemTypeEnum   designator;
   union {
      NWDPSelectionProblemEnum   standardProblem;
      NWDPObjectIdentifier       extendedProblem;
   } u;
} NWDPSelectionProblem, N_FAR *pNWDPSelectionProblem;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of problem:

0 NWDP_PROBLEM_TYPE_STANDARD
1 NWDP_PROBLEM_TYPE_EXTENDED

standardProblem

Specifies the type of standard problem:

0 NWDP_SELECTION_INVALID_ID
1 NWDP_SELECTION_UNKNOWN_ID
2 NWDP_SELECTION_OBJECT_EXISTS

extendedProblem

Specifies an OID describing the extended problem.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPSelectionProblem corresponds to ISO 10175-1 section 8.4.7.

NWDPServerListItem

Contains information known by the Service Registry Service about a service provider

Structure

```
typedef struct {
   NWDPText             serverName;
   NWDPSrsServerType    serverType;
   NWDPNetAddress       address;
   nuint                infoCount;
   pNWDPInfoItem        infoPtr;
} NWDPServerListItem, N_FAR *pNWDPServerListItem;
```

Header File

nwdp_srs.h

Fields

serverName

Specifies the advertised name of the server.

serverType

Specifies the type of server:

0 NWDP_SRS_SERVERTYPE_ALL
1 NWDP_SRS_SERVERTYPE_PA
2 NWDP_SRS_SERVERTYPE_NTS
3 NWDP_SRS_SERVERTYPE_RMS

address

Specifies the IPX address of the server.

infoCount

Specifies the number of NWDPInfoItem structures that are associated with the specified service. (Optional)

infoPtr

Points to the first NWDPInfoItem structure. (Optional)

Remarks

Applications can use the NWDPServiceListItem structure to discover information about the existence and location of services that are available through the Service Registry Service (SRS).

NWDPServiceError

Contains details of a service error within the Printer Agent or PSM

Structure

```
typedef struct {
   NWDPServiceProblem         problem;
   NWDPObjectIdentification   objectIdentification;
   pNWDPAttribute             attributeOptionPtr;
   nuint32                    libError;
   nuint32                    otherError;
   nuint32                    otherError2;
   pNWDPNameOrOid             messageOptionPtr;
} NWDPServiceError, N_FAR *pNWDPServiceError;
```

Header File

nwdp_att.h

Fields

problem

Specifies the problem.

objectIdentification

Specifies the object which has the problem.

attributeOptionPtr

Specifies the attribute of the object which has the problem. (Optional)

libError

Specifies the error returned to the NDPSM.NLM from usage of the DPLSV386.NLM library. etc.

otherError

Specifies the error returned to the NDPSM.NLM from usage of the DPLSV386.NLM library. etc.

libError2

Specifies the error returned to the NDPSM.NLM from usage of the DPLSV386.NLM library. etc.

messageOptionPtr

Specifies a message from the PSM to the caller. (Optional)

Remarks

NWDPServiceError corresponds to ISO 10175-1 section 8.4.7.

Print Service Group

NWDPServiceListItem

Contains a single service found on the Broker

Structure

```
typedef struct {
   nchar   serviceName[NWDP_BRK_MAX_NAME_SIZE];
   nbool   enableFlag;
} NWDPServiceListItem, N_FAR *pNWDPServiceListItem;
```

Header File

nwdp_brk.h

Fields

serviceName

Specifies the name of the supported service.

enableFlag

Specifies a Boolean that indicates if the service is enabled.

Remarks

You can receive the list of services and their current states by calling **NWDPBrkListServices** and providing a callback function.

NWDPServiceProblem

Contains details of a problem with a request to the Printer Agent or PSM

Structure

```
typedef struct {
   NWDPProblemTypeEnum   designator;
   union {
      NWDPServiceProblemEnum   standardProblem;
      NWDPObjectIdentifier     extendedProblem;
   } u;
} NWDPServiceProblem, N_FAR *pNWDPServiceProblem;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of standard problem:

0 NWDP_PROBLEM_TYPE_STANDARD
1 NWDP_PROBLEM_TYPE_EXTENDED

standardProblem

Specifies the type of standard problem:

0 NWDP_SERVICE_SERVER_BUSY
1 NWDP_SERVICE_SERVER_UNAVAILABLE
2 NWDP_SERVICE_OPERATION_COMPLEX
3 NWDP_SERVICE_RESOURCE_LIMIT
4 NWDP_SERVICE_UNCLASS_SERVER_ERR
5 NWDP_SERVICE_TOO_MANY_ITEM_LIST
6 NWDP_SERVICE_RESOURCE_NOT_AVAIL
7 NWDP_SERVICE_CANCEL_DOC_SUPPORT
8 NWDP_SERVICE_MODIFY_DOC_SUPPORT
9 NWDP_SERVICE_MULTI_DOC_SUPPORT
10 NWDP_SERVICE_PARM_VAL_SUPPORT
11 NWDP_SERVICE_INVALID_CHECKPOINT
12 NWDP_SERVICE_CONTINUATN_CONTEXT
13 NWDP_SERVICE_PAUSE_LIMIT_EXCEED
14 NWDP_SERVICE_UNSUPPORTED_OP

extendedProblem

Specifies an OID describing the extended problem.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPServiceProblem corresponds to ISO 10175-1 section 8.4.7.

NWDPSlittingSpec

Contains information about the slitting operation which is part of a finishing process

Structure

```
typedef struct {
   NWDPSlittingEnum   designator;
   union {
      NWDPNameOrOid        namedSlitting;
      NWDPSlitParameters   parameters;
   } u;
} NWDPSlittingSpec, N_FAR *pNWDPSlittingSpec;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of slitting specification:

0 NWDP_SLITTING_NAMED
1 NWDP_SLITTING_PARAMETERS

namedSlitting

Specifies a slitting operation; corresponds to ISO 10175-1: section 9.13.8.11.1.

parameters

Specifies a set of slitting parameters; corresponds to ISO 10175-1: section 9.13.8.11.2.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPSlittingSpec corresponds to ISO 10175-1: section 9.13.8.11.

NWDPSlitParameters

Contains parameters that control the slitting operation which is part of a finishing process

Structure

```
typedef struct {
   NWDPCommonParameters   common;
   NWDPLocations          headLocations;
} NWDPSlitParameters, N_FAR *pNWDPSlitParameters;
```

Header File

nwdp_att.h

Fields

common

Specifies generic finishing process parameters.

headLocations

Specifies a head location set; corresponds to ISO 10175-1: section 9.13.8.3.7.

Remarks

NWDPSlitParameters corresponds to ISO 10175-1: section 9.13.8.11.2.

NWDPStitchingSpec

Contains information about a stitching operation which is part of a finishing process

Structure

```
typedef struct {
   NWDPStitchingEnum   designator;
   union {
      NWDPNameOrOid          namedStitching;
      NWDPStitchParameters   parameters;
   } u;
} NWDPStitchingSpec, N_FAR *pNWDPStitchingSpec;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of stitching specification:

0 NWDP_STITCHING_NAMED
1 NWDP_STITCHING_PARAMETERS

namedStitching

Specifies a stitching operation; corresponds to ISO 10175-1: section 9.13.8.5.1.

parameters

Specifies a set of stitching parameters; corresponds to ISO 10175-1: section 9.13.8.5.2.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPStitchingSpec corresponds to ISO 10175-1: section 9.13.8.5.

NWDPStitchParameters

Contains parameters for a stitching operation

Structure

```
typedef struct {
   NWDPCommonParameters   common;
   pNWDPDimension         processOffsetOptionPtr;
   pNWDPLocations         headLocationsOptionPtr;
   NWDPNameOrOid          stitchType;
} NWDPStitchParameters, N_FAR *pNWDPStitchParameters;
```

Header File

nwdp_att.h

Fields

common

Specifies generic finishing process parameters.

processOffsetOptionPtr

Points to an offset from the reference edge; corresponds to ISO 10175-1, section 9.13.8.3.6. (Optional)

headLocationsOptionPtr

Points to an optional head location set; corresponds to ISO 10175-1: section 9.13.8.3.7.

stitchType

Specifies the type of stitch to apply as part of the finishing process; corresponds to ISO 10175-1: section 9.13.8.5.3.

Remarks

NWDPStitchParameters corresponds to ISO 10175-1: section 9.13.8.5.2.

NWDPSubStrings

Contains a substring matching criterion

Structure

```
typedef struct {
   NWDPObjectIdentifier          attributeId;
   NWDPSubstrMatchCriteriaEnum   matchCriteria;
   NWDPAttributeValue            initialValueOption;
   NWDPAttributeValueSet         anyValueSetOption;
   NWDPAttributeValue            finalValueOption;
} NWDPSubStrings, N_FAR *pNWDPSubStrings;
```

Header File

nwdp_att.h

Fields

attributeId

Specifies the attribute which contains the string to match against.

matchCriteria

Specifies the type of match criteria:

0 NWDP_MATCH_EXACT
1 NWDP_MATCH_CASE_INSENSITIVE
2 NWDP_MATCH_SAME_LETTER
3 NWDP_MATCH_APPROXIMATE

initialValueOption

Specifies a substring which must appear beginning at the first of the string.

anyValueSetOption

Specifies a set of substrings which are order-specific, but free floating which must all be present to match.

finalValueOption

Specifies a trailing string which must be the last characters of the string in the attribute in order to match.

Remarks

NWDPSubStrings corresponds to ISO 10175-1 section 6.4.5.1 (especially paragraph b).

NWDPText

Encapsulates various text forms (names or message strings) to be sent to the print service managers database

Structure

```
typedef struct
{
   nuint     itemCount;
   pnuint8   itemPtr;
} NWDPText, N_FAR *pNWDPText;
```

Header File

nwdp_att.h

Fields

itemCount

Specifies the number of bytes in the Unicode string.

itemPtr

Points to the NULL-terminated Unicode string in Intel (Little-Endian) format.

Remarks

Although *itemPtr* has type `pnuint8`, it is intended to point to a NULL-terminated Unicode string, so *itemCount* must include the NULL in the byte count.

NWDPTrimmingSpec

Contains information about a trimming operation which is part of a finishing process

Structure

```
typedef struct {
   NWDPTrimmingEnum   designator;
   union {
      NWDPNameOrOid        namedTrimming;
      NWDPTrimParameters   parameters;
   } u;
} NWDPTrimmingSpec, N_FAR *pNWDPTrimmingSpec;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of trimming specification:

0 NWDP_TRIMMING_NAMED
1 NWDP_TRIMMING_PARAMETERS

namedTrimming

Specifies a trimming operation; corresponds to ISO 10175-1: section 9.13.8.7.1.

parameters

Specifies a set of trimming parameters; corresponds to ISO 10175-1: section 9.13.8.7.2.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPTrimmingSpec corresponds to ISO 10175-1: section 9.13.8.7.

NWDPTrimParameters

Contains parameters for a trimming operation

Structure

```
typedef struct {
   NWDPCommonParameters   common;
   NWDPDimension          trimOffset;
   NWDPXYDimensions       trimDimensions;
} NWDPTrimParameters, N_FAR *pNWDPTrimParameters;
```

Header File

nwdp_att.h

Fields

common

Specifies generic finishing process parameters.

trimOffset

Specifies the offset value for the trimming operation. The default is zero.

trimDimensions

Specifies the dimensions for the trimming operation. The default value is the size of the reference-size parameter (corresponds to ISO 10175-1: section 9.13.8.3.4.)

Remarks

NWDPTrimParameters corresponds to ISO 10175-1: section 9.13.8.7.2.

NWDPTypedName

Contains an NDS name and two integers to match the Typed_Name_T structure found in NDS interfaces

Structure

```
typedef struct {
   NWDPText   name;
   nint32     level;
   nint32     interval;
} NWDPTypedName, N_FAR *pNWDPTypedName;
```

Header File

nwdp_att.h

Fields

name

Specifies an NDS object name.

level

Specifies an integer *level*.

interval

Specifies an integer *interval*.

Remarks

This structure is used so that the data can be transmitted to the Managed Object Database and/or to the NDS Directory database.

NWDPUpdateError

Contains details of an error which occurred while making a request of a Printer Agent or PSM

Structure

```
typedef struct {
   NWDPUpdateProblem          problem;
   NWDPObjectIdentification   objectIdentification;
   pNWDPNameOrOid             messageOptionPtr;
} NWDPUpdateError, N_FAR *pNWDPUpdateError;
```

Header File

nwdp_att.h

Fields

problem

Specifies the problem.

objectIdentification

Specifies the object with the problem.

messageOptionPtr

Specifies a message describing the problem. (Optional)

Remarks

NWDPUpdateError corresponds to ISO 10175-1 section 8.4.8.

NWDPUpdateProblem

Contains details of the update problem

Structure

```
typedef struct {
   NWDPProblemTypeEnum   designator;
   union {
      NWDPUpdateProblemEnum   standardProblem;
      NWDPObjectIdentifier    extendedProblem;
   } u;
} NWDPUpdateProblem, N_FAR *pNWDPUpdateProblem;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of problem:

0 NWDP_PROBLEM_TYPE_STANDARD
1 NWDP_PROBLEM_TYPE_EXTENDED

standardProblem

Specifies the type of standard problem:

0 NWDP_UPDATE_NO_MODS_ALLOWED
1 NWDP_UPDATE_INSUFFICIENT_RIGHTS
2 NWDP_UPDATE_PREV_OP_INCOMPLETE
3 NWDP_UPDATE_CANCEL_NOT_POSSIBLE

extendedProblem

Specifies an OID describing the extended problem.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPUpdateProblem corresponds to ISO 10175-1 section 8.4.8.

NWDPXYCardinalDimensions

Contains non-negative integer values that describes a rectangular area

Structure

```
typedef struct {
   nuint32   xDimension;
   nuint32   yDimension;
} NWDPXYCardinalDimensions, N_FAR *pNWDPXYCardinalDimensions;
```

Header File

nwdp_att.h

Fields

xDimension

Specifies the X dimension.

yDimension

Specifies the Y dimension.

Remarks

NWDPXYCardinalDimensions corresponds to ISO 10175-1 section 9.1.5.45.

NWDPXYDimensions

Contains a number pair and a tolerance that describes a rectangular area

Structure

```
typedef struct {
   NWDPXYDimensionsValue   value;
   NWDPRealOption          toleranceOption;
} NWDPXYDimensions, N_FAR *NWDPXYDimensions;
```

Header File

nwdp_att.h

Fields

value

Specifies the X and Y dimensions.

toleranceOption

Specifies a tolerance for the X and Y dimensions. (Optional)

Remarks

NWDPXYDimensions corresponds to ISO 10175-1 section 9.1.5.45.

NWDPXYDimensionsValue

Contains a number pair that describes a rectangular area

Structure

```
typedef struct {
   NWDPXYDimensionsValueEnum   designator;
   union {
      NWDPXYRealDimensions       realValue;
      NWDPNameOrOid              namedValue;
      NWDPXYCardinalDimensions   cardinalValue
   } u;
} NWDPXYDimensionsValue, N_FAR *pNWDPXYDimensionsValue;
```

Header File

nwdp_att.h

Fields

designator

Specifies the type of dimensions:

0 NWDP_DIM_XY_REAL
1 NWDP_DIM_XY_NAMED
2 NWDP_DIM_XY_CARDINAL

realValue

Specifies the XY dimensions as real values.

namedValue

Specifies the XY dimensions as a name or Oid.

cardinalValue

Specifies the XY dimensions as cardinal values.

Remarks

There is a one-to-one correspondence between the designator values and the fields within the union. NWDPXYDimensionsValue corresponds to ISO 10175-1 section 9.1.5.44.

NWDPXYRealDimensions

Contains a real number pair that describes a rectangular area

Structure

```
typedef struct {
   nreal64   xDimension;
   nreal64   yDimension;
} NWDPXYRealDimensions, N_FAR *pNWDPXYRealDimensions;
```

Header File

nwdp_att.h

Fields

xDimension

Specifies the X linear dimension in millimeters.

yDimension

Specifies the Y linear dimension in millimeters.

Remarks

NWDPXYRealDimensions corresponds to ISO 10175-1 section 9.1.5.45.

Print

Print: Guides

Print: Task Guide

Related Topics:

Parent Topic:

Print: Concept Guide

Related Topics:

Print Service Group

Parent Topic:

Print Overview

Print: Tasks

Capturing Print Jobs

NOTE: When an LPT device is captured, the NetWare redirector associates queue jobs with a queue. Any data the workstation sends to the captured LPT device is redirected to the server and placed in the queue.

1. **To initiate a capture to a print queue, call NWStartQueueCapture. This function redirects the specified LPT device. When a print job is sent to that LPT a capture file is opened and associated with a print queue. The queue is identified by connection ID and bindery object ID. In a similar fashion, NWStartFileCapture redirects the LPT device to a permanent capture file.**
2. **Print to the destination queue or file by sending data to the captured LPT device. You can use local print functions; the application printing need not be the application to initiate the capture.**
3. **To end a capture, call NWEndCapture. This function flushes any currently captured data to a server or a file and restores the LPT device's normal operation. Abort the redirection of an LPT device and discard any currently captured data by calling NWCancelCapture.**

An application may also flush or cancel the capture after it has been initiated. See Flushing a Capture.

Parent Topic:

Print Overview

Flushing a Capture

1. **You can flush data being captured to a queue on an LPT device by calling NWFlushCapture. This function leaves the LPT device captured and the capture flags unchanged. Data sent to the LPT device after the capture has been flushed is treated as a new job.**
2. **The NetWare shell can automatically flush a print queue job either according to a timeout value or by calling NWEndCapture. Two parameters control the automatic flushing of the device's capture file: the Flush Capture Timeout and the Flush Capture On Close flag. Both parameters are included in NWCAPTURE_FLAGS1.**

Print Service Group

Parent Topic:

Print Overview

Print: Concepts

Flush Capture On Close

The flush capture on close flag can be enabled or disabled. If enabled (0x00), the NetWare® server flushes the queue job when the application closes for any reason. This value applies to DOS and Windows environments only.

Parent Topic:

Flushing a Capture

Related Topics:

Flush Capture Timeout

Flush Capture Timeout

The flush capture timeout value starts counting down every time an application executes a print command (int 17h). When the timeout expires, the server flushes the file and queues it to a printer. If an application executes a second print command before the first timeout expires, the timeout starts over from the original value.

Each tick of the capture timeout is approximately one second. The range is from 0 to 65,535. The default timeout is 0x00, meaning no timeout. This value applies to DOS and Windows environments only.

Parent Topic:

Flushing a Capture

Related Topics:

Flush Capture On Close

Print Capture Functions

These functions control the capture status for an LPT device.

Function	Header	Comment
NWCancelCapture	nwprint.h	Cancels the capture of the specified LPT device without creating a print

		job for any data currently captured.
NWEndCapture	nwprint.h	Ends the capture of the specified LPT device.
NWFlushCapture	nwprint.h	Flushes the current data in the capture buffer to the print destination without closing the capture.
NWGetCaptureStatus	nwprint.h	Indicates whether the default LPT device is currently captured.
NWGetMaxPrinters	nwprint.h	Returns the number of LPT ports for which captures can be managed.
NWStartFileCapture	nwprint.h	Specifies a permanent network file that data will be captured to on the specified LPT device. Only one device can be captured to a file at a time. This function is available under DOS and Windows only.
NWStartQueueCapture	nwprint.h	Starts the capture of the specified LPT device to the specified NetWare® server and print queue.

Parent Topic:

Print Overview

Related Topics:

Print Status Functions

Print Job Capture Flags

Capture flags dictate how data sent to a captured LPT device should be printed. Once an LPT device has been captured, a pair of functions let you access the device's capture flags:

NWGetCaptureFlags reads the flags.

NWSetCaptureFlags modifies the flags.

Applications can both read and modify these capture flags:

Queue job control flags

Tab size

Number of copies

Print flags

Maximum lines

Maximum characters

Form name

Banner text

Flush capture timeout

Flush capture on close flag

NWCAPTURE_FLAGS1 contains this data.

Applications can read but can't modify these capture flags:

Connection ID

Queue ID

Setup string maximum length

Reset string maximum length

LPT capture flag

Timing out flag

In progress flag

Print job valid flag

Queue name

NWCAPTURE_FLAGS2 contains this data.

Related Topics:

Print Overview

Manipulating Capture Flags: Example

Print Introduction

Print functions integrate DOS, Windows*, and OS/2* workstations into the NetWare® printing environment. Through these functions, workstations redirect print jobs from local files or LPT devices to destinations on NetWare servers. Workstation applications may then submit print jobs for network printers or redirect jobs to permanent network files.

In versions of NetWare below 2.15C, the print server functionality is embedded in the server. NetWare 2.15C and higher provides the added functionality of print queues being serviced by print servers, which transfer

the jobs to network printers.

Although OS/2 and Netx allow only three captured LPT devices, the DOS Requester® allows up to nine LPT devices to be captured at the same time. The default value is 3 but can be changed by adding the statement NETWARE PRINTERS = x to the net.cfg file. If NETWARE PRINTERS = 0, the Requester does not load print.vlm. Moreover, each of a workstation's LPT devices can be captured separately. The DOS Requester allocates memory for each LPT device captured.

For a description of structures and other data definitions relating to this chapter, see Print: Structures.

Parent Topic:

Print Overview

Print Status Functions

These functions read and modify the workstation's capture flags and return printer status.

Function	Header	Comment
NWGetBannerUserName	nwprint.h	Returns the value for the banner name capture flag. The same value is applied to all LPT devices.
NWGetCaptureFlags	nwprint.h	Returns the print job capture flags for the specified LPT device.
NWGetPrinterStatus	nwprint.h	Returns the current status of the printer as a PRINTER_STATUS structure [NetWare 2.15].
NWGetPrinterStrings	nwprint.h	Returns the printer setup string and printer reset string for the specified LPT device. The size of the string buffers is returned by **NWGetCaptureFlags**.
NWSetBannerUserName	nwprint.h	Modifies the value for the banner name capture flag. The same value is applied to all LPT devices.
NWSetCaptureFlags	nwprint.h	Modifies the print job capture flags for the specified LPT device. Note only the data defined in the NWCAPTURE_FLAGS1 structure can be modified.
NWSetPrinterStrings	nwprint.h	Assigns values to the printer setup string and printer reset string for the specified LPT device. The maximum

		length of the buffers are returned by **NWGetCaptureFlags**.

Parent Topic:

Print Overview

Related Topics:

Print Capture Functions

Print: Functions

NWCancelCapture

Cancels the capture for the specified LPT device and aborts any print jobs being processed

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows* 3.1, Windows NT*, Windows*95

Service: Print

Syntax

```
#include <nwprint.h>
or
#include <nwcalls. h>

NWCCODE NWAPI NWCancelCapture (
   nuint8   LPTDevice);
```

Pascal Syntax

```
#include <nwprint.inc>

Function NWCancelCapture
  (LPTDevice : nuint8
) : NWCCODE;
```

Parameters

LPTDevice

(IN) Specifies the LPT device number.

Return Values

These are common return values; see Return Values for more information.

0x0000	SUCCESSFUL
0x8836	INVALID_PARAMETER

Remarks

NWCancelCapture ends the redirection of a specified LPT device. If the specified LPT device was in the process of capturing to a file, **NWCancelCapture** aborts and deletes the print queue job entry.

LPTDevice can have the following values:

```
1    LPT1
2    LPT2
3    LPT3
```

Under VLM and OS/2, *LPTDevice* can have up to nine values (1-9).

To start another capture of the LPT device, **NWStartQueueCapture** or **NWStartFileCapture** must be called again.

NCP Calls

None

See Also

NWEndCapture, **NWFlushCapture**, **NWGetCaptureFlags**, **NWGetCaptureStatus** , **NWSetCaptureFlags**, **NWStartFileCapture**, **NWStartQueueCapture**

NWEndCapture

Flushes the data to the queue and ends the capture for the specified LPT device

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print

Syntax

```
#include<nwprint.h>
or
#include <nwcalls.h>

NWCCODE NWAPI NWEndCapture (
   nuint8   LPTDevice);
```

Pascal Syntax

```
#include <nwprint.inc>

Function NWEndCapture
  (LPTDevice : nuint8
) : NWCCODE;
```

Parameters

LPTDevice

(IN) Specifies the LPT device number.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	SUCCESSFUL

Remarks

Once an application calls **NWEndCapture**, the workstation shell no longer traps data sent to the specified LPT device.

NWEndCapture sends a message to the server that the capture is complete.

NWEndCapture also closes the capture file (if one exists) associated with the specified LPT device, allowing a printer to print the file.

Under Netx, *LPTDevice* can have the following values:

```
1   LPT1
2   LPT2
3   LPT3
```

Under VLM and OS/2, *LPTDevice* can have up to nine values (1-9).

For DOS/Windows, *LPTCaptureFlag*, *printQueueFlag*, and *printJobValidFlag*(fields in NWCAPTURE_FLAGSR0) are cleared. (These flags do not exist for OS/2.)

NCP Calls

None

See Also

NWCancelCapture, **NWFlushCapture**, **NWGetCaptureFlags**, **NWGetCaptureStatus**, **NWSetCaptureFlags**, **NWStartFileCapture**, **NWStartQueueCapture**

NWFlushCapture

Flushes the buffers and ends the queue job but maintains the port redirection

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print

Syntax

```
#include<nwprint.h>
or
#include <nwcalls.h>

NWCCODE NWAPI NWFlushCapture (
   nuint8   LPTDevice);
```

Pascal Syntax

```
#include <nwprint.inc>

Function NWFlushCapture
  (LPTDevice : nuint8
) : NWCCODE;
```

Parameters

LPTDevice

(IN) Specifies the number of the specified LPT device.

Return Values

These are common return values; see Return Values for more information.

0x0000	SUCCESSFUL

Remarks

When **NWFlushCapture** is executed, the specified LPT device remains captured and redirected to the NetWare® server. If the appplication sends more data to the specified LPT devices, the data is redirected to a new capture file on the NetWare server for printing.

NWFlushCapture clears *captureInProgress* and *printJobValid*(fields in FlagBufferStruct). *LPTCaptureFlag* and *printQueueFlag* remain set.

Under Netx, *LPTDevice* can have the following values:

```
1   LPT1
2   LPT2
3   LPT3
```

Under VLM and OS/2, *LPTDevice* can have up to nine values (1-9).

An application can only call **NWFlushCapture** after calling **NWStartFileCapture** or **NWStartQueueCapture**.

NCP Calls

None

See Also

NWStartFileCapture, **NWStartQueueCapture**

NWGetBannerUserName

Returns the user name printed on the banner pages for print jobs sent to any LPT device

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print

Syntax

```
#include <nwprint.h>
or
#include <nwcalls.h>

NWCCODE NWAPI NWGetBannerUserName (
   pnstr8   userName);
```

Pascal Syntax

```
#include <nwprint.inc>

Function NWGetBannerUserName
  (username : pnstr8
) : NWCCODE;
```

Parameters

userName

(OUT) Points to the banner user name (13 characters maximum, including NULL).

Return Values

These are common return values; see Return Values for more information.

0x0000	SUCCESSFUL

Remarks

An application cannot specify a different user name for each LPT device on a user's workstation.

NCP Calls

None

See Also

NWGetCaptureFlags, **NWSetBannerUserName**, **NWSetCaptureFlags**

NWGetCaptureFlags

Allows the application to get capture information
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print

Syntax

```
#include <nwprint.h>
or
#include <nwcalls. h>

NWCCODE NWAPI NWGetCaptureFlags (
   nuint                    LPTDevice,
   NWCAPTURE_FLAGS1 N_FAR   *captureFlags1,
   NWCAPTURE_FLAGS2 N_FAR   *captureFlags2);
```

Pascal Syntax

```
#include <nwprint.inc>

Function NWGetCaptureFlags
  (LPTDevice : nuint8;
   Var captureFlags1 : NWCAPTURE_FLAGS1;
   Var captureFlags2 : NWCAPTURE_FLAGS2
) : NWCCODE;
```

Parameters

LPTDevice

(IN) Specifies the LPT device number.

captureFlags1

(OUT) Points to NWCAPTURE_FLAGS1 containing capture flags for the specified LPT device (optional).

captureFlags2

(OUT) Points to NWCAPTURE_FLAGS2 containing capture flags for the specified LPT device (optional).

Return Values

These are common return values; see Return Values for more information.

0x0000	SUCCESSFUL

Remarks

Under Netx, *LPTDevice* can have the following values:

```
1   LPT1
2   LPT2
3   LPT3
```

Under VLM and OS/2, *LPTDevice* can have up to nine values (1-9).

Under VLM and Client32, the connection ID contained in the *connID* field of the NWCAPTURE_FLAGS2 structure needs to be converted to a connection handle to maintain backward compatibility with NWCALLS. If the *connID* field specifies an invalid connection handle, NETX will return the default connection handle since NETX does not recognize connection references.

Before calling **NWGetCaptureFlags**, set the NWCAPTURE_FLAGS1 structure fields to zero before calling **NWGetCaptureFlags**. If **NWGetCaptureFlags** is then called for an uncaptured port, both structures will not contain valid data from the last time **NWGetCaptureFlags** was called.

NWCAPTURE_FLAGS1 contains flags that can be set and NWCAPTURE_FLAGS2 contains read-only information.

NCP Calls

None

See Also

NWGetCaptureStatus, **NWSetCaptureFlags**, **NWStartFileCapture**, **NWStartQueueCapture**

NWGetCaptureStatus

Returns whether a specific device is currently captured
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print

Syntax

```
#include <nwprint.h>
or
#include <nwcalls. h>

NWCCODE NWAPI NWGetCaptureStatus (
   nuint8   LPTDevice);
```

Pascal Syntax

```
#include <nwprint.inc>

Function NWGetCaptureStatus
  (LPTDevice : nuint8
) : NWCCODE;
```

Parameters

LPTDevice

(IN) Specifies the LPT device number.

Return Values

These are common return values; see Return Values for more information.

0x0000	LPT device is not being captured
0x00FF	LPT device is currently being captured
any other value	LPT device is not getting captured

Remarks

Under Netx, *LPTDevice* can have the following values:

```
1    LPT1
2    LPT2
3    LPT3
```

Under VLM and OS/2, *LPTDevice* can have up to nine values (1-9).

NCP Calls

None

See Also

NWStartFileCapture, **NWStartQueueCapture**

NWGetMaxPrinters

Returns the total number of physical and logical LPT ports that can be redirected to a queue

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print

Syntax

```
#include <nwprint.h>
or
#include <nwcalls.h>

NWCCODE NWFAR NWPASCAL NWGetMaxPrinters (
   pnuint16   numPrinters);
```

Pascal Syntax

```
#include <nwprint.inc>

Function NWGetMaxPrinters
  (numPrinters : pnuint16
) : NWCCODE;
```

Parameters

numPrinters

(OUT) Points to the number of printers that can be captured.

Return Values

These are common return values; see Return Values for more information.

0x0000	SUCCESSFUL
0x8848	BAD_VLM_ERROR

Remarks

NWGetMaxPrinters returns the number of LPT ports for which captures can be managed:

```
OS/2              9
```

```
DOS shell          3
DOS VLM shell    0-9
```

To configure the DOS VLM shell, set NetWare Printers = x in net.cfg, where x is an integer 0-9.

Memory is allocated for each specified LPT port.

NCP Calls

None

NWGetPrinterStrings

Returns the printer setup strings and the printer reset strings
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print

Syntax

```
#include <nwprint.h>
or
#include <nwcalls.h>

NWCCODE NWAPI NWGetPrinterStrings (
   nuint8     LPTDevice,
   pnuint16   setupStringLen,
   pnstr8     setupString,
   pnuint16   resetStringLen,
   pnstr8     resetString);
```

Pascal Syntax

```
#include <nwprint.inc>

Function NWGetPrinterStrings
  (LPTDevice : nuint8;
   setupStringLen : pnuint16;
   setupString : pnstr8;
   resetStringLen : pnuint16;
   resetString : pnstr8
) : NWCCODE;
```

Parameters

LPTDevice

(IN) Specifies the LPT device number.

setupStringLen

(IN/OUT) Points to the length of the buffer *setupString* upon input. Points to the length of the actual *setupString* returned (optional) upon output.

setupString

(OUT) Points to a buffer in which to put *setupString* (optional).

resetStringLen

(IN/OUT) Points to the length of the buffer *resetString* upon input. Points to the length of the actual *resetString* returned (optional).

resetString

(OUT) Points to a buffer in which to put *resetString* (optional).

Return Values

These are common return values; see Return Values for more information.

0x0000	SUCCESSFUL

Remarks

A buffer the size of *setupStringMaxLen* and *resetStringMaxLen* (fields in NWCAPTURE_FLAGS2) must be provided for **NWGetPrinterStrings**. These lengths can be retrieved by calling **NWGetCaptureFlags**. The actual string lengths are returned in *setupStringLen* and *resetStringLen*. The strings are not NULL-terminated and may contain NULLs.

If set to NULL, *setupString* is not filled and no value is returned in *setupStringLen*. If set to NULL, *resetString* is not filled and no value is returned in *resetStringLen.*

Under Netx, *LPTDevice* can have the following values:

```
1   LPT1
2   LPT2
3   LPT3
```

Under VLM and OS/2, *LPTDevice* can have up to nine values (1-9).

NCP Calls

0x2222 17 06 Get Printer Status

See Also

NWGetCaptureFlags, **NWSetPrinterStrings**

NWSetBannerUserName

Sets the user name printed on banner pages for print jobs sent to LPT devices

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print

Syntax

```
#include <nwprint.h>
or
#include <nwcalls. h>

NWCCODE NWAPI NWSetBannerUserName (
   pnstr8   userName);
```

Pascal Syntax

```
#include <nwprint.inc>

Function NWSetBannerUserName
  (username : pnstr8
) : NWCCODE;
```

Parameters

userName

(IN) Points to the banner user name (13 characters maximum, including NULL).

Return Values

These are common return values; see Return Values for more information.

0x0000	SUCCESSFUL

Remarks

An application cannot specify a different user name for each LPT device on a users workstation.

NCP Calls

Print Service Group

None

NWSetCaptureFlags

Allows the application to set flags that pertain to the device redirection and print job information using NWCAPTURE_FLAGS1

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print

Syntax

```
#include <nwprint.h>
or
#include <nwcalls. h>

NWCCODE NWAPI NWSetCaptureFlags (
   NWCONN_HANDLE             conn,
   nuint8                    LPTDevice,
   NWCAPTURE_FLAGS1 N_FAR  *captureFlags1);
```

Pascal Syntax

```
#include <nwprint.inc>

Function NWSetCaptureFlags
  (conn : NWCONN_HANDLE;
   LPTDevice : nuint8;
   Var captureFlags1 : NWCAPTURE_FLAGS1
) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

LPTDevice

(IN) Specifies the number of the specified LPT device.

captureFlags1

(IN) Points to NWCAPTURE_FLAGS1 containing the capture flags to be set for the specified LPT device.

Return Values

These are common return values; see Return Values for more information.

0x0000	SUCCESSFUL

Remarks

Under Netx, *LPTDevice* can have the following values:

```
1    LPT1
2    LPT2
3    LPT3
```

Under VLM and OS/2, *LPTDevice* can have up to nine values (1-9).

NCP Calls

None

See Also

NWGetCaptureFlags , **NWGetCaptureStatus**, **NWStartFileCapture**, **NWStartQueueCapture**

NWSetPrinterStrings

Sets the printer setup string and the printer reset string
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print

Syntax

```
#include <nwprint.h>
or
#include <nwcalls.h>

NWCCODE NWAPI NWSetPrinterStrings (
   nuint8    LPTDevice,
   nuint16   setupStringLen,
   pnstr8    setupString,
   nuint16   resetStringLen,
   pnstr8    resetString);
```

Pascal Syntax

```
#include <nwprint.inc>

Function NWSetPrinterStrings
  (LPTDevice : nuint8;
   setupStringLen : nuint16;
   setupString : pnstr8;
   resetStringLen : nuint16;
   resetString : pnstr8
) : NWCCODE;
```

Parameters

LPTDevice

(IN) Specifies the LPT device number.

setupStringLen

(IN) Specifies the length of the data in setup string; *setupStringLen* is not used if *setupString* is set to NULL.

setupString

(IN) Points to a buffer containing the setup string (optional).

resetStringLen

(IN) Specifies the length of the data in *resetString*; *resetStringLen* is not used if *resetString* is set to NULL

resetString

(IN) Points to a buffer containing the reset string (optional).

Return Values

These are common return values; see Return Values for more information.

0x0000	SUCCESSFUL

Remarks

The strings must not exceed the lengths specified in *setupStringMaxLen* and *resetStringMaxLen* (fields in NWCAPTURE_FLAGS2). These lengths can be retrieved by calling **NWGetCaptureFlags**.

Under Netx, *LPTDevice* can have the following values:

```
1   LPT1
2   LPT2
3   LPT3
```

Under VLM and OS/2, *LPTDevice* can have up to nine values (1-9).

Under Windows NT, the strings are dynamically allocated and the values returned in the NWCAPTURE_FLAGS2 structure should be ignored.

NCP Calls

None

See Also

NWGetCaptureFlags, **NWGetCaptureStatus**, **NWGetPrinterStrings**

NWStartFileCapture

Lets a DOS or Windows application capture data to a permanent network file

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, Windows NT, Windows 3.1, Windows95

Service: Print

Syntax

```
#include <nwprint.h>
or
#include <nwcalls.h>

NWCCODE NWAPI NWStartFileCapture (
   NWCONN_HANDLE   conn,
   nuint8          LPTDevice,
   NWDIR_HANDLE    dirHandle,
   pnstr8          filePath);
```

Pascal Syntax

```
#include <nwprint.inc>

Function NWStartFileCapture
  (conn : NWCONN_HANDLE;
   LPTDevice : nuint8;
   dirhandle : NWDIR_HANDLE;
   filePath : pnstr8
) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle if NETX.VLM is running.

LPTDevice

(IN) Specifies the LPT device number.

dirHandle

(IN) Specifies the index to a file path; *dirHandle* can be 0 if *filePath* is an absolute path including the volume name.

filePath

(IN) Points to an absolute path, or a path relative to the directory handle, specifying the file to be created or used as a permanent capture file.

Return Values

These are common return values; see Return Values for more information.

0x0000	SUCCESSFUL

Remarks

NWStartFileCapture does not queue the file for printing.

NOTE: NWStartFileCapture is not available for OS/2.

conn is only valid when NETX.VLM is loaded. Otherwise, *conn* is ignored.

The requesting workstation must have Read, Write, and Create rights to the directory where the file is created.

To call **NWStartFileCapture**, an application should:

1. Call **NWStartFileCapture**.
2. Send data to the captured LPT device.
3. End the capture by calling **NWCancelCapture**.

IMPORTANT: The workstation shell traps data sent to the captured LPT device and redirects the data to the file that the application specified in **NWStartFileCapture**. Finally, the application should cancel the capture of the LPT device to close the capture file.

If the default capture flags are used in capturing the file, the file will automatically close upon receipt of the data. If you wish the file to stay open, set *flushCaptureOnClose* to off (1) in the NWCAPTURE_FLAGS1 structure. The file will then remain open for more data input until **NWCancelCapture** is called. *flushCaptureOnClose* will default to on (0).

WARNING: If NWEndCapture is called in any application that also calls NWStartFileCapture, 0x884C DEVICE_NOT_REDIRECTED will be returned.

An application can reopen the file and capture additional data. However, reopening the file moves the file pointer back to the beginning of the file, and the previously captured data is overwritten from the beginning of the file.

Under Netx, *LPTDevice* can have the following values:

```
1   LPT1
2   LPT2
```

```
3   LPT3
```

Under VLM and OS/2, *LPTDevice* can have up to nine values (1-9).

dirHandle is an index number (1 to 255) that points to a volume or a directory on the NetWare server. Drive handles are recorded in the directory handle table that a NetWare server maintains for each logged-in workstation.

filePath can identify a full or a partial directory path. The partial directory path must combine with the path pointed to by DirHandle to create a full path. A full directory path defines a volume or a directory on a given NetWare server in the format:

```
VOLUME:DIRECTORY\ . . . \DIRECTORY
```

A partial directory path specifies at least a directory, and possibly one or more parent directories.

A partial file path includes a file name and (optionally) one or more antecedent directory names. A file name can be 1 to 8 characters long and can also include an extension of 1 to 3 characters. All letters must appear in upper case.

Applications can combine a directory handle and a partial file path to identify a file.

NCP Calls

0x2222 22 01 Get Directory Path

NWStartQueueCapture

Redirects a specified device to a queue
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print

Syntax

```
#include <nwprint.h>
or
#include <nwcalls. h>

NWCCODE NWAPI NWStartQueueCapture (
   NWCONN_HANDLE   conn,
   nuint8          LPTDevice,
   nuint32         queueID,
   pnstr8          queueName);
```

Pascal Syntax

```
#include <nwprint.inc>

Function NWStartQueueCapture
  (conn : NWCONN_HANDLE;
   LPTDevice : nuint8;
   queueID : nuint32;
   queueName : pnstr8
) : NWCCODE;
```

Parameters

conn

(IN) Specifies the NetWare server connection handle.

LPTDevice

(IN) Specifies the LPT device number.

queueID

(IN) Specifies the bindery object ID of the print queue on the target server.

queueName

(IN) Points to the print queue name (up to 65 characters).

Return Values

These are common return values; see Return Values for more

information.

0x0000	SUCCESSFUL
0x89FC	NO_SUCH_OBJECT

Remarks

Call **NWStartQueueCapture** to change the current capture information values.

When trying to capture to a different server, ensure you pass in the connection handle for the server to whose queue you want to attach. Otherwise, **NWStartQueueCapture** will return NO_SUCH_OBJECT.

When an application sends data to the captured LPT device, the workstation shell traps the data and redirects it to a capture file on a NetWare server.

The NetWare server creates the capture file and an associated print queue job number when the first character is sent to the redirected LPT device. The NetWare server also places a hold on the print queue job number while data is captured to the file. Once the application closes the capture file with **NWEndCapture** or **NWFlushCapture**, a printer servicing the queue can print the file.

Under Netx, *LPTDevice* can have the following values:

```
1    LPT1
2    LPT2
3    LPT3
```

Under VLM and OS/2, *LPTDevice* can have up to nine values (1-9).

NCP Calls

0x2222 23 17 Get File Server Information
0x2222 23 102 Read Queue Current Status
0x2222 23 125 Read Queue Current Status

Print: Structures

CaptureFlagsStruct

Service: Print
Defined In: nwprint.h

Structure

```
typedef struct
{
    nuint8    status;
    nuint8    flags;
    nuint8    tabSize;
    nuint8    serverPrinter;
    nuint8    numberCopies;
    nuint8    formType;
    nuint8    reserved;
    nuint8    bannerText[13];
    nuint8    reserved2;
    nuint8    localLPTDevice;
    nuint16   captureTimeOutCount;
    nuint8    captureOnDeviceClose;
} CaptureFlagsStruct;
```

Pascal Structure

```
Defined in nwprint.inc

  CaptureFlagsStruct = Record
    status : nuint8;
    flags : nuint8;
    tabSize : nuint8;
    serverPrinter : nuint8;
    numberCopies : nuint8;
    formType : nuint8;
    reserved : nuint8;
    bannerText : Array[0..12] Of nuint8;
    reserved2 : nuint8;
    localLPTDevice : nuint8;
    captureTimeOutCount : nuint16;
    captureOnDeviceClose : nuint8
  End;
```

Fields

status

flags

tabSize

Indicates the tab size which is a value between 1 and 18 inclusive. The default setting is 0x08.

serverPrinter

numberCopies

Indicates the number of copies of the captured file the printer prints (The maximum number of copies is 255 for netx.com, and 65536 for OS/2 and the DOS Requester). The default setting is 0x0001. If *numberCopies* is 0x0000, nothing prints.

formType

Indicates the type of form (0 to 255) a user must mount in the printer to print files captured to the LPT device. If the form currently mounted in the printer differs from the form type returned in this field, the NetWare server console displays a message instructing the console operator to mount the correct form. The default form is 0x0000.

reserved

bannerText

Indicates a 13-byte string containing the name appearing on the bottom half of a banner page. All letters are upper case.

reserved2

localLPTDevice

captureTimeOutCount

captureOnDeviceClose

FlagBufferStruct

Service: Print
Defined In: nwprint.h

Structure

```
typedef struct
{
   nuint8    status;
   nuint8    flags;
   nuint8    tabSize;
   nuint8    serverPrinter;
   nuint8    numberCopies;
   nuint8    formType;
   nuint8    reserved;
   nuint8    bannerText[13];
   nuint8    reserved2;
   nuint8    localLPTDevice;
   nuint16   captureTimeOutCount;
   nuint8    captureOnDeviceClose;
   nuint16   maxLines;
   nuint16   maxChars;
   nuint8    formName[13];
   nuint8    LPTCaptureFlag;
   nuint8    fileCaptureFlag;
   nuint8    timingOutFlag;
   pnstr8    printerSetupBuffAddr;
   pnstr8    printerResetBuffAddr;
   nuint8    connID;
   nuint8    captureInProgress;
   nuint8    printQueueFlag;
   nuint8    printJobValid;
   nuint32   queueID;
   nuint16   printJobNumber;
} FlagBufferStruct;
```

Pascal Structure

```
Defined in nwprint.inc

 FlagBufferStruct = Record
    status : nuint8;
    flags : nuint8;
    tabSize : nuint8;
    serverPrinter : nuint8;
    numberCopies : nuint8;
    formType : nuint8;
    reserved : nuint8;
```

```
  bannerText : Array[0..12] Of nuint8;
  reserved2 : nuint8;
  localLPTDevice : nuint8;
  captureTimeOutCount : nuint16;
  captureOnDeviceClose : nuint8;
  maxLines : nuint16;
  maxChars : nuint16;
  formName : Array[0..12] Of nuint8;
  LPTCaptureFlag : nuint8;
  fileCaptureFlag : nuint8;
  timingOutFlag : nuint8;
  printerSetupBuffAddr : pnstr8;
  printerResetBuffAddr : pnstr8;
  connID : nuint8;
  captureInProgress : nuint8;
  printQueueFlag : nuint8;
  printJobValid : nuint8;
  queueID : nuint32;
  printJobNumber : nuint16
End;
```

Fields

status

flags

tabSize

Indicates the tab size which is a value between 1 and 18 inclusive. The default setting is 0x08.

serverPrinter

numberCopies

Indicates the number of copies of the captured file the printer prints (The maximum number of copies is 255 for netx.com, and 65536 for OS/2 and the DOS Requester). The default setting is 0x0001. If *numberCopies* is 0x0000, nothing prints.

formType

Indicates the type of form (0 to 255) a user must mount in the printer to print files captured to the LPT device. If the form currently mounted in the printer differs from the form type returned in this field, the NetWare server console displays a message instructing the console operator to mount the correct form. The default form is 0x0000.

reserved

bannerText

Indicates a 13-byte string containing the name appearing on the bottom half of a banner page. All letters are upper case.

reserved2

localLPTDevice

captureTimeOutCount

captureOnDeviceClose

maxLines

Indicates the maximum lines per page.

maxChars

Indicates the maximum characters per line.

formName

Indicates the name of the form a user must mount in the printer to print files captured to the LPT device. If the form currently mounted in the printer differs from the form name returned in this field, the NetWare server console displays a message instructing the console operator to mount the correct form.

LPTCaptureFlag

Indicates whether the LPT device is captured. (Set = 0xFF, clear = 0x00.) This value applies to DOS and Windows environments only.

fileCaptureFlag

Indicates a capture filename has been specified. All captured data is sent to the file and not to a print queue. (Set = 0xFF, clear = 0x00.) This value applies to DOS and Windows environments only.

timingOutFlag

Indicates *flushCaptureTimeout* is decrementing. The flag is cleared if the timeout isn't decrementing. (Set = 0xFF, clear = 0x00.) This value applies to DOS and Windows environments only.

printerSetupBuffAddr

printerResetBuffAddr

connID

Indicates the connection handle identifying the connection.

captureInProgress

printQueueFlag

printJobValid

queueID

Indicates the bindery ID of the target queue.

printJobNumber

Indicates a number used to identify the job.

NWCAPTURE_FLAGS1

Contains parameters that applications can both read and modify
Service: Print
Defined In: nwprint.h

Structure

```
typedef struct
{
   nuint8    jobDescription[50];
   nuint8    jobControlFlags;
   nuint8    tabSize;
   nuint16   numCopies;
   nuint16   printFlags;
   nuint16   maxLines;
   nuint16   maxChars;
   nuint8    formName[13];
   nuint8    reserved[9];
   nuint16   formType;
   nuint8    bannerText[13];
   nuint8    reserved2;
   nuint16   flushCaptureTimeout;
   nuint8    flushCaptureOnClose;
} NWCAPTURE_FLAGS1;
```

Pascal Structure

```
Defined in nwprint.inc

NWCAPTURE_FLAGSRW = Record
    jobDescription : Array[0..49] Of nuint8;
    jobControlFlags : nuint8;
    tabSize : nuint8;
    numCopies : nuint16;
    printFlags : nuint16;
    maxLines : nuint16;
    maxChars : nuint16;
    formName : Array[0..12] Of nuint8;
    reserved : Array[0..8] Of nuint8;
    formType : nuint16;
    bannerText : Array[0..12] Of nuint8;
    reserved2 : nuint8;
    flushCaptureTimeout : nuint16;
    flushCaptureOnClose : nuint8
  End;
```

Fields

jobDescription

Indicates the null-terminated ASCII description of the contents or purpose of the job. The NetWare DOS Requester™ uses only 13 bytes of this member. This member is used only by the DOS Requester (12 bytes and NULL) and OS/2* (full buffer).

jobControlFlags

Indicates the set of queue job control flags affecting the way a queue server processes a queue job. This member is used only by the DOS Requester and OS/2. Returns 0 under DOS/Windows. Under OS/2 *jobControlFlags* is defined as 0x0400 (print interrupted capture), and bits 0, 1, and 2 must be 0.

tabSize

numCopies

Indicates the number of copies of the captured file the printer prints (The maximum number of copies is 255 for netx.com, and 65536 for OS/2 and the DOS Requester). The default setting is 0x0001. If *numCopies* is 0x0000, nothing prints.

printFlags

maxLines

Indicates the maximum lines per page.

maxChars

Indicates the maximum characters per line.

formName

Indicates the name of the form a user must mount in the printer to print files captured to the LPT device. If the form currently mounted in the printer differs from the form name returned in this field, the NetWare server console displays a message instructing the console operator to mount the correct form.

reserved

formType

Indicates the type of form (0 to 255) a user must mount in the printer to print files captured to the LPT device. If the form currently mounted in the printer differs from the form type returned in this field, the NetWare server console displays a message instructing the console operator to mount the correct form. The default form is 0x0000.

bannerText

Indicates a 13-byte string containing the name appearing on the bottom half of a banner page. All letters are upper case.

reserved2

flushCaptureTimeout

Indicates a value (0 to 3,640) that starts counting down every time an application executes a print command (int 17h). When the timeout

expires, the server flushes the capture file and queues it at a printer. If an application executes a second print command before the first timeout expires, the timeout starts over from the original value. Each tick of the capture timeout is approximately one second. The range is 1 to 1000. The default timeout is 0x0000---no timeout. (Not valid in OS/2.)

flushCaptureOnClose

Indicates the server will flush the capture file when the application ends the capture of the default LPT device. (Default is 0x0000.) Any other value means disabled. (Not valid in OS/2.)

Remarks

Under Windows NT, *printFlags*, *bannerText*, *flushCaptureTimeout*, and *flushCaptureOnClose* are not set.

NWCAPTURE_FLAGS2

Contains information applications can read but not modify
Service: Print
Defined In: nwprint.h

Structure

```
typedef struct
{
   NWCONN_HANDLE   connID;
   nuint32         queueID;
   nuint16         setupStringMaxLen;
   nuint16         resetStringMaxLen;
   nuint8          LPTCaptureFlag;
   nuint8          fileCaptureFlag;
   nuint8          timingOutFlag;
   nuint8          inProgress;
   nuint8          printQueueFlag;
   nuint8          printJobValid;
   nstr8           queueName[65];
} NWCAPTURE_FLAGS2;
```

Pascal Structure

```
Defined in nwprint.inc

    NWCAPTURE_FLAGSRO = Record
    connID : NWCONN_HANDLE;
    queueID : nuint32;
    setupStringMaxLen : nuint16;
    resetStringMaxLen : nuint16;
    LPTCaptureFlag : nuint8;
    fileCaptureFlag : nuint8;
    timingOutFlag : nuint8;
    inProgress : nuint8;
    printQueueFlag : nuint8;
    printJobValid : nuint8;
    queueName : Array[0..64] Of nstr8
  End;
```

Fields

connID

Indicates the connection handle identifying the connection.

queueID

Indicates the bindery ID of the target queue.

setupStringMaxLen

Indicates the size of the buffer for setup strings. The string prepares the printer to receive and correctly print the capture file. The buffer size can be specified by adding the statement

```
PRINT HEADER =
```

to the net.cfg file.

resetStringMaxLen

Indicates the size of the buffer for print reset strings. This string resets the printer after the capture file is printed. The buffer size can be specified by adding the statement

```
PRINT TAIL =
```

to the net.cfg file.

LPTCaptureFlag

Indicates whether the LPT device is captured. (Set = 0xFF, clear = 0x00.) This value applies to DOS and Windows environments only.

fileCaptureFlag

Indicates a capture filename has been specified. All captured data is sent to the file and not to a print queue. (Set = 0xFF, clear = 0x00.) This value applies to DOS and Windows environments only.

timingOutFlag

Indicates *flushCaptureTimeout* is decrementing. The flag is cleared if the timeout isn't decrementing. (Set = 0xFF, clear = 0x00.) This value applies to DOS and Windows environments only.

inProgress

Indicates the first character of the print job was sent to the specified LPT device. The flag is cleared when the capture file is closed. (Set = 0xFF, clear = 0x00.) This value applies to DOS and Windows environments

printQueueFlag

Indicates the server queues and numbers the job. (Set = 0xFF, clear = 0x00.) This value applies to DOS and Windows environments only.

printJobValid

Indicates the capture file is open to receive characters and cleared when the capture file is closed. (Set = 0xFF, clear = 0x00.) This value applies to DOS and Windows environments only.

queueName

Indicates the VLM™ application only.

NWPipeStruct

Service: Print
Defined In: nwprint.h

Structure

```
typedef struct
{
   nuint16            fwCommand;
   nuint16            idSession;
   nuint32            idQueue;
   nuint16            idConnection;
   nuint16            idDevice;
   nuint16            fwMode;
   nuint16            fwScope;
   nuint16            cbBufferLength;
   nuint8             fbValidBuffer;
   SpoolFlagsStruct   nwsSpoolFlags;
   nuint8             szBannerUserName[13];
   nuint16            rc;
} NWPipeStruct;
```

Fields

fwCommand

idSessionidQueue

idQueue

idConnection

idDevice

fwMode

fwScope

cbBufferLength

fbValidBuffer

nwsSpoolFlags

szBannerUserName

rc

NWPrintJobStruct

Service: Print
Defined In: nwprint.h

Structure

```
typedef struct
{
   nuint32   clientStation;
   nuint32   clientTask;
   nuint32   clientID;
   nuint32   targetServerID;
   nuint8    targetExecutionTime[6];
   nuint8    jobEntryTime[6];
   nuint32   jobNumber;
   nuint16   formType;
   nuint16   jobPosition;
   nuint16   jobControlFlags;
   nuint8    jobFileName[14];
   nuint32   jobFileHandle;
   nuint32   servicingServerStation;
   nuint32   servicingServerTask;
   nuint32   servicingServerID;
   nuint8    jobDescription[50];
   nuint8    clientJobInfoVer;
   nuint8    tabSize;
   nuint16   numberCopies;
   nuint16   printFlags;
   nuint16   maxLines;
   nuint16   maxChars;
   nuint8    formName[16];
   nuint8    reserved[6];
   nuint8    bannerUserName[13];
   nuint8    bannerUserName[13];
   nuint8    bannerHeaderFileName[14];
   nuint8    filePathName[80];
} NWPrintJobStruct;
```

Pascal Structure

```
Defined in nyprint.inc

  NWPrintJobStruct = Record
    clientStation : nuint32;
    clientTask : nuint32;
    clientID : nuint32;
    targetServerID : nuint32;
    targetExecutionTime : Array[0..5] Of nuint8;
```

```
    jobEntryTime : Array[0..5] Of nuint8;
    jobNumber : nuint32;
    formType : nuint16;
    jobPosition : nuint16;
    jobControlFlags : nuint16;
    jobFileName : Array[0..13] Of nuint8;
    jobFileHandle : nuint32;
    servicingServerStation : nuint32;
    servicingServerTask : nuint32;
    servicingServerID : nuint32;
    jobDescription : Array[0..49] Of nuint8;
    clientJobInfoVer : nuint8;
    tabSize : nuint8;
    numberCopies : nuint16;
    printFlags : nuint16;
    maxLines : nuint16;
    maxChars : nuint16;
    formName : Array[0..15] Of nuint8;
    reserved : Array[0..5] Of nuint8;
    bannerUserName : Array[0..12] Of nuint8;
    bannerFileName : Array[0..12] Of nuint8;
    bannerHeaderFileName : Array[0..13] Of nuint8;
    filePathName : Array[0..79] Of nuint8
  End;
```

Fields

clientStation

Indicates the client submitting the job to the queue.

clientTask

Indicates the task number the station was performing when it placed the job in the queue.

clientID

clientID

targetServerID

Indicates the server ID of the queue server servicing the job. If this field is set to 0xFFFFFFFF, any queue server can service the job. If the specified queue server is not attached to the queue, QMS removes the job from the queue.

targetExecutionTime

Indicates the earliest time the job can be serviced. The time is in the format: year, month, day, hour, minute, second (YYMMDDHHMMSS). If this field is set to 0xFFFFFFFFFFFF, the job is serviced at the first opportunity.

jobEntryTime

Indicates the time the client submitted the job to the queue.

jobNumber

Indicates a number used to identify the job.

formType

Indicates the type of form (0 to 255) a user must mount in the printer to print files captured to the LPT device. If the form currently mounted in the printer differs from the form type returned in this field, the NetWare server console displays a message instructing the console operator to mount the correct form. The default form is 0x0000.

jobPosition

Indicates the job's position in the queue. The first job is assigned position 1, the next is assigned position 2, and so on. As jobs are removed from the queue, this number changes to reflect the updated position of the queue job.

jobControlFlags

Indicates the set of queue job control flags affecting the way a queue server processes a queue job. This member is used only by the DOS Requester and OS/2. Returns 0 under DOS/Windows. Under OS/2 *jobControlFlags* is defined as 0x0400 (print interrupted capture), and bits 0, 1, and 2 must be 0.

jobFileName

Indicates the name of the job file (in DOS 8.3 format) created by the QMS. The name of the file is in the form Q$XXXX.YYY.

jobFileHandle

Indicates the handle of the job file created by the QMS. The name of the file is in the form Q$XXXX.YYY. It is normally not used by the client. Queue Services provide the client with a file handle specific to the local file system (DOS or OS/2) for accessing the job.

servicingServerStation

Indicates the station number of the queue server servicing the job. If the job isn't being processed, this value is undefined.

servicingServerTask

Indicates the task number of the queue server servicing the job. If the job isn't being processed, this value is undefined.

servicingServerID

Indicates the server. If the job isn't being processed, this value is undefined.

jobDescription

Indicates the NULL-terminated ASCII text description of the content or purpose of a job. QMS displays this text as part of the job description when users or operators examine a queue.

clientJobInfoVer

Indicates additional information passed to the job server by the client. The information can take any format agreed upon by both the client and the job server. This field must be supplied by the client creating

the queue job.

tabSize

Indicates the tab size which is a value between 1 and 18 inclusive. The default setting is 0x08.

numberCopies

Indicates the number of copies of the captured file the printer prints (The maximum number of copies is 255 for netx.com, and 65536 for OS/2 and the DOS Requester). The default setting is 0x0001. If *numberCopies* is 0x0000, nothing prints.

printFlags

Release Job indicates the print job is released for printing if the capture is interrupted by a loss of connection time to the server.

Suppress Form Feed indicates the print service suppresses automatic form feed after the print job is printed.

Text File indicates that tab size and other printer control sequences are interpreted by the print service. If not set, the job is interpreted as a byte stream.

Print Banner indicates the print service precedes the print job with a banner page.

maxLines

Indicates the maximum lines per page.

maxChars

Indicates the maximum characters per line.

formName

Indicates the name of the form a user must mount in the printer to print files captured to the LPT device. If the form currently mounted in the printer differs from the form name returned in this field, the NetWare server console displays a message instructing the console operator to mount the correct form.

reserved

bannerUserName

bannerUserName

bannerHeaderFileName

filePathName

Indicates a full or partial directory path. A full directory path defines a volume or a directory on a given file server.

PRINTER_STATUS

Reads printer status information for NetWare 2.15 printing
Service: Print
Defined In: nwprint.h

Structure

```
typedef struct
{
   nuint8   printerHalted;
   nuint8   printerOffline;
   nuint8   currentFormType;
   nuint8   redirectedPrinter;
} PRINTER_STATUS;
```

Pascal Structure

```
Defined in nwprint.inc

  PRINTER_STATUS = Record
    printerHalted : nuint8;
    printerOffline : nuint8;
    currentFormType : nuint8;
    redirectedPrinter : nuint8
  End;
```

Fields

printerHalted

Indicates if the printer has been halted from the system console: Set = 0x89FF, clear = 0x00.

printerOffline

Indicates if the printer is offline: Set = 0x89FF, clear = 0x00.

currentFormType

Indicates the number of the form mounted on the printer.

redirectedPrinter

Indicates the target printer. May be a value other than the target printer if printing has been redirected at the system console and jobs are being rerouted to another queue and printer.

PrintJobStruct

Service: Print
Defined In: nwprint.h

Structure

```
typedef struct
{
   nuint8    clientStation;
   nuint8    clientTask;
   nuint32   clientID;
   nuint32   targetServerID;
   nuint8    targetExecutionTime[6];
   nuint8    jobEntryTime[6];
   nuint16   jobNumber;
   nuint16   formType;
   nuint8    jobPosition;
   nuint8    jobControlFlags;
   nuint8    jobFileName[14];
   nuint8    jobFileHandle[6];
   nuint8    servicingServerStation;
   nuint8    servicingServerTask;
   nuint32   servicingServerID;
   nuint8    jobDescription[50];
   nuint8    clientJobInfoVer;
   nuint8    tabSize;
   nuint16   numberCopies;
   nuint16   printFlags;
   nuint16   maxLines;
   nuint16   maxChars;
   nuint8    formName[16];
   nuint8    reserved[6];
   nuint8    bannerUserName[13];
   nuint8    bannerFileName[13];
   nuint8    bannerHeaderFileName[14];
   nuint8    filePathName[80];
} PrintJobStruct;
```

Pasca Structure

```
Defined in nwprint.inc

PrintJobStruct = Record
  PrintJobStruct = Record
    clientStation : nuint8;
    clientTask : nuint8;
    clientID : nuint32;
    targetServerID : nuint32;
```

```
    targetExecutionTime : Array[0..5] Of nuint8;
    jobEntryTime : Array[0..5] Of nuint8;
    jobNumber : nuint16;
    formType : nuint16;
    jobPosition : nuint8;
    jobControlFlags : nuint8;
    jobFileName : Array[0..13] Of nuint8;
    jobFileHandle : Array[0..5] Of nuint8;
    servicingServerStation : nuint8;
    servicingServerTask : nuint8;
    servicingServerID : nuint32;
    jobDescription : Array[0..49] Of nuint8;
    clientJobInfoVer : nuint8;
    tabSize : nuint8;
    numberCopies : nuint16;
    printFlags : nuint16;
    maxLines : nuint16;
    maxChars : nuint16;
    formName : Array[0..15] Of nuint8;
    reserved : Array[0..5] Of nuint8;   (* must be set to zeros *)
    bannerUserName : Array[0..12] Of nuint8;
    bannerFileName : Array[0..12] Of nuint8;
    bannerHeaderFileName : Array[0..13] Of nuint8;
    filePathName : Array[0..79] Of nuint8
  End;
```

Fields

clientStation

Indicates the client submitting the job to the queue.

clientTask

Indicates the task number the station was performing when it placed the job in the queue.

clientID

Indicates the object ID of the person submitting the job.

targetServerID

Indicates the server ID of the queue server servicing the job. If this field is set to 0xFFFFFFFF, any queue server can service the job. If the specified queue server is not attached to the queue, QMS removes the job from the queue.

targetExecutionTime

Indicates the earliest time that the job can be serviced. The time is in this format: year, month, day, hour, minute, second. If this field is set to 0xFFFFFFFFFFFF, the job is serviced at the first opportunity.

jobEntryTime

Indicates the time the client submitted the job to the queue.

jobNumber

Indicates a number used to identify the job.

formType

Indicates the type of form (0 to 255) a user must mount in the printer to print files captured to the LPT device. If the form currently mounted in the printer differs from the form type returned in this field, the NetWare server console displays a message instructing the console operator to mount the correct form. The default form is 0x0000.

jobPosition

jobPosition

jobControlFlags

jobControlFlags

jobFileName

Indicates the name of the job file (in DOS 8.3 format) created by the QMS. The name of the file is in the form Q$XXXX.YYY.

jobFileHandle

Indicates the handle of the job file created by the QMS. The name of the file is in the form Q$XXXX.YYY. It is normally not used by the client. Queue Services provide the client with a file handle specific to the local file system (DOS or OS/2) for accessing the job.

servicingServerStation

Indicates the station number of the queue server servicing the job. If the job isn't being processed, this value is undefined.

servicingServerTask

Indicates the task number of the queue server servicing the job. If the job isn't being processed, this value is undefined.

servicingServerID

Indicates the server. If the job isn't being processed, this value is undefined.

jobDescription

Indicates the NULL-terminated ASCII text description of the content or purpose of a job. QMS displays this text as part of the job description when users or operators examine a queue.

clientJobInfoVer

Indicates additional information passed to the job server by the client. The information can take any format agreed upon by both the client and the job server. This field must be supplied by the client creating the queue job.

tabSize

Indicates the tab size which is a value between 1 and 18 inclusive. The default setting is 0x08.

numberCopies

Indicates the number of copies (0 to 255) of the capture file the printer prints. The default setting is 0x01. If *numberCopies* is 0x00, nothing will be printed.

printFlags

Release Job indicates the print job is released for printing if the capture is interrupted by a loss of connection time to the server.

Suppress Form Feed indicates the print service suppresses automatic form feed after the print job is printed.

Text File indicates that tab size and other printer control sequences are interpreted by the print service. If not set, the job is interpreted as a byte stream.

Print Banner indicates the print service precedes the print job with a banner page.

maxLines

Indicates the naximum lines per page. The default is 66.

maxChars

Indicates the maximum characters per line. The default is 132.

formName

Indicates the name of the form that a user must mount in the printer to print files captured to the default LPT device. If the form that is currently mounted in the printer differs from the form name returned in this field, the file server console displays a message instructing the console operator to mount the correct form.

reserved

bannerUserName

bannerFileName

bannerHeaderFileName

filePathName

Indicates a full or partial directory path. A full directory path defines a volume or a directory on a given file server.

SpoolFlagsStruct

Service: Print
Defined In: nwprint.h

Structure

```
typedef struct
{
   nuint32          targetServerID;
   nuint8           targetExecutionTime[6];
   nuint8           jobDescription[50];
   nuint8           jobControlFlags;
   nuint8           tabSize;
   nuint16          numberCopies;
   nuint16          printFlags;
   nuint16          maxLines;
   nuint16          maxChars;
   nuint8           formName[16];
   nuint8           reserved1[6];
   nuint16          formType;
   nuint8           bannerFileName[13];
   nuint8           reserved2;
   NWCONN_HANDLE    connID;
   nuint32          queueID;
   nuint16          setupStringMaxLength;
   nuint16          resetStringMaxLength;
} SpoolFlagsStruct;
```

Fields

targetServerID

Indicates the server ID of the queue server servicing the job. If this field is set to 0xFFFFFFFF, any queue server can service the job. If the specified queue server is not attached to the queue, QMS removes the job from the queue.

targetExecutionTime

Indicates the earliest time the job can be serviced. The time is in the format: year, month, day, hour, minute, second (YYMMDDHHMMSS). If this field is set to 0xFFFFFFFFFFFF, the job is serviced at the first opportunity.

jobDescription

Indicates the NULL-terminated ASCII text description of the content or purpose of a job. QMS displays this text as part of the job description when users or operators examine a queue.

jobControlFlags

Indicates the set of queue job control flags affecting the way a queue server processes a queue job. This member is used only by the DOS Requester and OS/2. Returns 0 under DOS/Windows. Under OS/2 *jobControlFlags* is defined as 0x0400 (print interrupted capture), and bits 0, 1, and 2 must be 0.

tabSize

Indicates the tab size which is a value between 1 and 18 inclusive. The default setting is 0x08.

numberCopies

Indicates the number of copies of the captured file the printer prints (The maximum number of copies is 255 for netx.com, and 65536 for OS/2 and the DOS Requester). The default setting is 0x0001. If *numberCopies* is 0x0000, nothing prints.

printFlags

Release Job indicates the print job is released for printing if the capture is interrupted by a loss of connection time to the server.

Suppress Form Feed indicates the print service suppresses automatic form feed after the print job is printed.

Text File indicates that tab size and other printer control sequences are interpreted by the print service. If not set, the job is interpreted as a byte stream.

Print Banner indicates the print service precedes the print job with a banner page.

maxLines

Indicates the maximum lines per page.

maxChars

Indicates the maximum characters per line.

reserved1

formType

Indicates the type of form (0 to 255) a user must mount in the printer to print files captured to the LPT device. If the form currently mounted in the printer differs from the form type returned in this field, the NetWare server console displays a message instructing the console operator to mount the correct form. The default form is 0x0000.

bannerFileName

reserved2

connID

Indicates the connection handle identifying the connection.

queueID

Indicates the bindery ID of the target queue.

setupStringMaxLength

resetStringMaxLength

Print Server

Print Server: Functions

NWPSCfgAddNServer

Adds a NetWare® server to a list of those served by a print server
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows* 3.1, Windows NT*, Windows*95
Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgAddNServer (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *pServerName,
   char NWFAR   *nServerName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE_PRE_40 or NWPS_BINDERY_SERVICE.

connID

(IN) Specifies the connection identifier.

pServerName

(IN) Points to the name of the print server.

nServerName

(IN) Points to the name of the NetWare server to add.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

For the print server to service queues on another NetWare server, the

NetWare server must be specified and saved by calling **NWPSCfgAddNServer**.

Because Directory Services does not need NetWare server names, **NWPSCfgAddNServer** fails if *connType* equals NWPS_DIRECTORY_SERVICE.

NCP Calls

None

See Also

NWPSCfgDeleteNServer, **NWPSCfgGetFirstNServer**, **NWPSCfgGetNextNServer**, **NWPSCfgEndNextNServer**, **NWPSCfgVerifyNServer**, **NWPSCfgGetFirstPrintServer**, **NWPSCfgGetNextPrintServer**, **NWPSCfgEndNextPrintServer**, **NWPSCfgVerifyPrintServer**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSCfgAddPrinter

Adds a printer to the list of printers of a defined print server and adds several default attributes to the printer

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgAddPrinter
  (WORD          connType,
   DWORD         connID,
   char NWFAR   *pServerName,
   char NWFAR   *printerName,
   WORD NWFAR   *printerNumber);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

pServerName

(IN) Points to the name of the print server.

printerName

(IN) Points to the name of the printer to add.

printerNumber

(IN) Points to the number of the printer to add.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services

Remarks

printerNumber is used for the print server printer attribute and can be NULL or a pointer to -1 if the caller does not care what that number is. A number is assigned and returned if not NULL. *printerNumber* is required for Bindery identification.

In NWPS_BINDERY_SERVICE, the list of operators contains SUPERVISOR. In NWPS_DIRECTORY_SERVICE, the list of operators contains the current user.

The default attributes follow:

```
NWPS_ATTR_OPER   = SUPERVISOR (Bindery Services)
NWPS_ATTR_OPER   = current user (Directory Services)
NWPS_ATTR_STAT   = NWPS_NOT_CONNECTED
NWPS_ATTR_CONF   = LPT1
NWPS_ATTR_NOTIFY = job owner
NWPS_ATTR_OWNER  = current user
```

NCP Calls

None

See Also

NWPSCfgAddPrinterAttr, **NWPSCfgDeletePrinter**, **NWPSCfgGetFirstPrinter**, **NWPSCfgGetNextPrinter**, **NWPSCfgEndNextPrinter**, **NWPSCfgVerifyPrinter**, **NWPSCfgGetFirstPrintServer**, **NWPSCfgGetNextPrintServer**, **NWPSCfgEndNextPrintServer**, **NWPSCfgVerifyPrintServer**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSCfgAddPrinterAttr

Adds an attribute to an existing printer
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
 #include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgAddPrinterAttr (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *pServerName,
   char NWFAR   *printerName,
   WORD          attrID,
   void NWFAR   *attrValue);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE_PRE_40, NWPS_BINDERY_SERVICE, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

pServerName

(IN) Points to the name of the print server.

printerName

(IN) Points to the name of the printer.

attrID

(IN) Specifies the print services attribute identifier.

attrValue

(IN) Points to the attribute buffer.

Return Values

0x0000	Successful
-1	General Error

other values	Bindery or Directory Services Errors

Remarks

The type of the buffer pointed to by *attrValue* depends on the value of *attrID*. If *attrID* is not recognized, an error is returned.

If *attrValue* is single-valued, the new value overwrites the old value. If *attrValue* is multi-valued, the new attribute value is added to the list.

Legal attribute identifiers and values for the printer configuration follow:

Bindery Identifier	Directory Identifier	Type	Multi Valued
None	NWPS_ATTR_ACL	Object_ACL_T	Yes
None	NWPS_ATTR_CART	char[ ]	Yes
NWPS_ATTR_CN	NWPS_ATTR_CN	char[ ]	No
NWPS_ATTR_CONF	NWPS_ATTR_CONF	Octet_String_T	No
NWPS_ATTR_DESC	NWPS_ATTR_DESC	char[ ]	No
NWPS_ATTR_DQUEUE	NWPS_ATTR_DQUEUE	char[ ]	No
None	NWPS_ATTR_HOST_DEV	char[ ]	No
None	NWPS_ATTR_MEMORY	Integer_T	No
None	NWPS_ATTR_NADD	Net_Address_T	Yes
None	NWPS_ATTR_NADD_REST	Net_Address_T	Yes
NWPS_ATTR_NOTIFY	NWPS_ATTR_NOTIFY	NWPS_Typed_Name	Yes
NWPS_ATTR_OPER	NWPS_ATTR_OPER	NWPS_Typed_Name	Yes
NWPS_ATTR_OWNER	NWPS_ATTR_OWNER	char[ ]	Yes
None	NWPS_ATTR_PAGE	char[ ]	Yes
NWPS_ATTR_P	NWPS_ATTR_PRI	Typed_Name	No

NWPS_ATTR_PRINT_SER	NWPS_ATTR_PRINT_SER	Typed_Name_T	No
NWPS_ATTR_QUEUE	NWPS_ATTR_QUEUE	Typed_Name_T	Yes
None	NWPS_ATTR_SEE_ALSO	char[ ]	Yes
None	NWPS_ATTR_SERIAL	char[ ]	Yes
NWPS_ATTR_STAT	NWPS_ATTR_STAT	Integer_T	No
None	NWPS_ATTR_TYPE (faces)	char[ ]	Yes

NOTE: char[ ] is a NULL-terminated character string.

NWPS_ATTR_STAT values follow:

```
0 = NWPS_PSTAT_JOB_WAIT
1 = NWPS_PSTAT_FORM_WAIT
2 = NWPS_PSTAT_PRINTING
3 = NWPS_PSTAT_PAUSED
4 = NWPS_PSTAT_STOPPED
5 = NWPS_PSTAT_MARK_EJECT
6 = NWPS_PSTAT_READY_TO_DOWN
7 = NWPS_PSTAT_NOT_CONNECTED
8 = NWPS_PSTAT_PRIVATE
```

NCP Calls

None

See Also

NWPSCfgAddPrinter, **NWPSCfgDeletePrinterAttr**, **NWPSCfgGetFirstPrinterAttr**, **NWPSCfgGetNextPrinterAttr**, **NWPSCfgEndNextPrinterAttr**, **NWPSCfgModifyPrinterAttr**, **NWPSCfgGetFirstPrintServer**, **NWPSCfgGetNextPrintServer**, **NWPSCfgEndNextPrintServer**, **NWPSCfgVerifyPrintServer**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSCfgAddPrintQueue

Creates a print queue using a default operator and user attributes
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
 #include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgAddPrintQueue (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *queueName,
   char NWFAR   *volumeName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

queueName

(IN) Points to the name of the print queue.

volumeName

(IN) Points to the volume name where the queue directory is to be created.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

In NWPS_BINDERY_SERVICE mode, the volume name is automatically assigned; therefore, *volumeName* is ignored. In NWPS_DIRECTORY_SERVICE mode, *volumeName* must be a valid directory volume object.

After the queue is created, it can be added to a printer with **NWPSCfgAddPrinterAttr**, using either NWPS_ATTR_DQUEUE or NWPS_ATTR_QUEUE. When a queue is created, it is assigned a default NWPS_ATTR_QUE_DIR. This applies to both NWPS_DIRECTORY_SERVICE and NWPS_BINDERY_SERVICE.

In NWPS_BINDERY_SERVICE, the operator is set to the currently logged-in user and the user is set to EVERYONE. If the user group EVERYONE does not exist, the currently logged-in user is added to the list of users explicitly. A local server is assumed.

In NWPS_DIRECTORY_SERVICE, the list of operators is set to the current user and the list of users is set to the organizational unit. If there is a group EVERYONE under the organizational unit, it is also added to the list of users for compatibility with Bindery emulation.

The default value for NWPS_ATTR_VOLUME is the volume name. The default value for NWPS_ATTR_HOST_SER is the name of the NetWare server where the volume is found.

NCP Calls

None

See Also

NWPSCfgAddPrintQueueAttr, **NWPSCfgDeletePrintQueue**, **NWPSCfgGetFirstPrintQueue**, **NWPSCfgGetNextPrintQueue**, **NWPSCfgEndNextPrintQueue**, **NWPSCfgVerifyPrintQueue**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSCfgAddPrintQueueAttr

Adds or creates a queue attribute value
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgAddPrintQueueAttr (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *queueName,
   WORD          attrID,
   void NWFAR   *attrValue);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

queueName

(IN) Points to the name of the print queue.

attrID

(IN) Specifies the print services attribute identifier.

attrValue

(IN) Points to the attribute value buffer.

Return Values

0x0000	Successful
-1	Failure or unknown attribute name
other values	Attribute number identifier

Remarks

The type of the buffer pointed to by *attrValue* depends on the value of *attrID*. If *attrID* is not recognized, an error is returned.

Legal attribute identifiers for the printer's queue follow:

Bindery Identifier	Directory Identifier	Type	Multi Valued
None	NWPS_ATTR_ACL	Object_ACL_T	Yes
NWPS_ATTR_CN	NWPS_ATTR_CN	char[ ]	No
NWPS_ATTR_DESC	NWPS_ATTR_DESC	char[ ]	No
NWPS_ATTR_DEVICE	NWPS_ATTR_DEVICE	char[ ]	Yes
NWPS_ATTR_HOST_RES	NWPS_ATTR_HOST_RES	char[ ]	No
NWPS_ATTR_HOST_SER	NWPS_ATTR_HOST_SER	char[ ]	No
None	NWPS_ATTR_NADD	Net_Address_T	Yes
NWPS_ATTR_OPER	NWPS_ATTR_OPER	NWPS_Typed_Name	Yes
NWPS_ATTR_QUE_DIR	NWPS_ATTR_QUE_DIR	NWPS_Typed_Name	Yes
None	NWPS_ATTR_SEE_ALSO	char[ ]	Yes
NWPS_ATTR_SERVER	NWPS_ATTR_SERVER	char[ ]	Yes
NWPS_ATTR_USER	NWPS_ATTR_USER	NWPS_Typed_Name	Yes
NWPS_ATTR_VOLUME	NWPS_ATTR_VOLUME	char[ ]	No

NOTE: char[ ] is a NULL-terminated character string.

NCP Calls

None

See Also

NWPSCfgDeletePrintQueueAttr, **NWPSCfgGetFirstPrintQueueAttr**, **NWPSCfgGetNextPrintQueueAttr**, **NWPSCfgEndNextPrintQueueAttr**, **NWPSCfgModifyPrintQueueAttr**, **NWPSCfgAddPrintQueue**, **NWPSCfgGetFirstPrintQueue**, **NWPSCfgGetNextPrintQueue**, **NWPSCfgEndNextPrintQueue**, **NWPSCfgVerifyPrintQueue**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSCfgAddPrintServer

Adds a print server to a Bindery-based NetWare server or to a Directory Services context, and creates a password

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgAddPrintServer (
   WORD          connType,
   DWORD         connID,
   char NWFAR  *pServerName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

pServerName

(IN) Points to the name of the print server to add.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

NWPSCfgAddPrintServer also adds the following attributes to the print server object:

```
NWPS_ATTR_OPER    =    SUPERVISOR (Bindery Services)
NWPS_ATTR_OPER    =    current user (Directory Services)
NWPS_ATTR_USER    =    EVERYONE (Bindery Services)
NWPS_ATTR_USER    =    current least significant organizational unit (Dir
NWPS_ATTR_STAT    =    NWPS_DOWN (both Bindery and Directory Services)
```

NCP Calls

None

See Also

NWPSCfgAddPrintServerAttr, **NWPSCfgDeletePrintServer**, **NWPSCfgGetFirstPrintServer**, **NWPSCfgGetNextPrintServer**, **NWPSCfgEndNextPrintServer**, **NWPSCfgVerifyPrintServer**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSCfgAddPrintServerAttr

Adds an attribute to an existing print server
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgAddPrintServerAttr (
   WORD          connType,
   WORD          connID,
   char NWFAR   *pServerName,
   WORD          attrID,
   void NWFAR   *attrValue);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

pServerName

(IN) Points to the name of the print server.

attrID

(IN) Specifies the print server attribute identifier.

attrValue

(IN) Points to the attribute value.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

The type of the buffer pointed to by *attrValue* depends on the value of *attrID*. If *attrID* is not recognized, an error is returned.

If *attrValue* is single-valued, the new value overwrites the old value. If *attrValue* is multi-valued, the new attribute value is added to the list.

Legal attribute identifiers and values for the print server follow:

Bindery Identifier	Directory Identifier	Type	Multi Valued
None	NWPS_ATTR_ACL	Object_ACL_T	Yes
NWPS_ATTR_CN	NWPS_ATTR_CN	char[ ]	No
NWPS_ATTR_DESC	NWPS_ATTR_DESC	char[ ]	No
None	NWPS_ATTR_HOST_DEV	char[ ]	No
None	NWPS_ATTR_NADD	Net_Address_T	Yes
NWPS_ATTR_OPER	NWPS_ATTR_OPER	NWPS_Typed_Name	Yes
NWPS_ATTR_PRINTER	NWPS_ATTR_PRINTER	Typed_Name_T	Yes
None	NWPS_ATTR_PRIV_KEY	Octet_String_T	No
None	NWPS_ATTR_PUBL_KEY	Octet_String_T	No
NWPS_ATTR_SAP	NWPS_ATTR_SAP	char[ ]	No
None	NWPS_ATTR_SEE_ALSO	char[ ]	Yes
NWPS_ATTR_STAT	NWPS_ATTR_STAT	Integer_T	No
NWPS_ATTR_USER	NWPS_ATTR_USER	char[ ]	Yes
NWPS_ATTR_VERS	NWPS_ATTR_VERS	char[ ]	No

NOTE: char[ ] is a NULL-terminated character string.

NWPS_ATTR_STAT values follow:

```
0 = NWPS_RUNNING
1 = NWPS_GOING_DOWN
2 = NWPS_DOWN
3 = NWPS_INITIALIZING
```

NCP Calls

None

See Also

NWPSCfgAddPrintServer, **NWPSCfgDeletePrintServerAttr**, **NWPSCfgGetFirstPrintServerAttr**, **NWPSCfgGetNextPrintServerAttr**, **NWPSCfgEndNextPrintServerAttr**, **NWPSCfgModifyPrintServerAttr**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSCfgAttrNameToNumber

Converts the attribute name to its print services attribute identifier
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgAttrNameToNumber (
   char NWFAR  *attributeName);
```

Parameters

attributeName

(IN) Points to the name of the attribute to convert.

Return Values

0x0000	Successful
-1	Failure or unknown attribute name
other values	Attribute number identifier

Remarks

attributeName is a NULL-terminated ASCII string.

NCP Calls

None

See Also

NWPSCfgAttrNumberToName

NWPSCfgAttrNumberToName

Using a print service attribute ID, returns a pointer to the print services attribute name

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

char NWFAR *far pascal NWPSCfgAttrNumberToName (
   WORD   attributeID);
```

Parameters

attributeID

(IN) Specifies the print server attribute identifier.

Return Values

NULL	Failure or invalid attribute identifier
pointer	Attribute name (NULL-terminated ASCII string)

NCP Calls

None

See Also

NWPSCfgAttrNameToNumber

NWPSCfgDeleteNServer

Deletes a NetWare server and its associated objects from the list of NetWare servers serviced by a print server

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgDeleteNServer (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *pServerName,
   char NWFAR   *nServerName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE or NWPS_BINDERY_SERVICE_PRE_40.

connID

(IN) Specifies the connection identifier.

pServerName

(IN) Points to the name of the print server.

nServerName

(IN) Points to the name of the NetWare server to delete.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

NWPSCfgDeleteNServer applies only if *connType* is NWPS_BINDERY_SERVICE.

NCP Calls

None

See Also

NWPSCfgAddNServer, **NWPSCfgGetFirstNServer**, **NWPSCfgGetNextNServer**, **NWPSCfgEndNextNServer**, **NWPSCfgVerifyNServer**, **NWPSCfgGetFirstPrintServer**, **NWPSCfgGetNextPrintServer**, **NWPSCfgEndNextPrintServer**, **NWPSCfgVerifyPrintServer**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSCfgDeletePrinter

Deletes a previously defined printer from the print server's list of printers
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgDeletePrinter (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *pServerName,
   char NWFAR   *printerName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

pServerName

(IN) Points to the name of the print server.

printerName

(IN) Points to the name of the printer to delete.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

In NWPS_DIRECTORY_SERVICE mode, the printer is also removed from the printer attribute of the print server and from the Directory itself.

Both *pServerName* and *printerName* are NULL-terminated ASCII strings.

NCP Calls

None

See Also

NWPSCfgDeletePrinterAttr, **NWPSCfgGetFirstPrinter**, **NWPSCfgGetNextPrinter**, **NWPSCfgEndNextPrinter**, **NWPSCfgVerifyPrinter**, **NWPSCfgGetFirstPrintServer**, **NWPSCfgGetNextPrintServer**, **NWPSCfgEndNextPrintServer**, **NWPSCfgVerifyPrintServer**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSCfgDeletePrinterAttr

Deletes a printer attribute from the Bindery or Directory Services
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgDeletePrinterAttr (
   WORD          connType,
   DWORD         connID,
   char NWFAR    pServerName,
   char NWFAR   *printerName,
   WORD          attrID,
   void NWFAR   *attrValue);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

pServerName

(IN) Points to the name of the print server.

printerName

(IN) Points to the name of the printer.

attrID

(IN) Specifies the print services attribute identifier.

attrValue

(IN) Points to the attribute buffer.

Return Values

0x0000	Successful
-1	General Error

other values	Bindery or Directory Services Errors

Remarks

The type of the buffer pointed to by *attrValue* depends on the value of *attrID*. If *attrID* is not recognized, an error is returned.

See **NWPSCfgAddPrinterAttr** for a list of legal attributes and corresponding types.

NCP Calls

None

See Also

NWPSCfgAddPrinterAttr, **NWPSCfgGetFirstPrinterAttr**, **NWPSCfgGetNextPrinterAttr**, **NWPSCfgEndNextPrinterAttr**, **NWPSCfgModifyPrinterAttr**, **NWPSCfgGetFirstPrinter**, **NWPSCfgGetNextPrinter**, **NWPSCfgEndNextPrinter**, **NWPSCfgVerifyPrinter**, **NWPSCfgGetFirstPrintServer**, **NWPSCfgGetNextPrintServer**, **NWPSCfgEndNextPrintServer**, **NWPSCfgVerifyPrintServer**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSCfgDeletePrintQueue

Deletes a print queue from a printer object, from a print server object, and from the Bindery or Directory

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgDeletePrintQueue (
   WORD          connType,
   DWORD         connID,
   char NWFAR  *queueName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

queueName

(IN) Points to the name of the queue to delete.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

Any reference to the queue is also removed from printers and print servers within the same context as the print queue.

NCP Calls

None

See Also

NWPSCfgAddPrintQueue, **NWPSCfgDeletePrintQueueAttr**, **NWPSCfgGetFirstPrintQueue**, **NWPSCfgGetNextPrintQueue**, **NWPSCfgEndNextPrintQueue**, **NWPSCfgVerifyPrintQueue**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSCfgDeletePrintQueueAttr

Deletes a printer queue attribute, but keeps the queue
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgDeletePrintQueueAttr (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *queueName,
   WORD          attrID,
   void NWFAR   *attrValue);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

queueName

(IN) Points to the queue name to modify.

attrID

(IN) Specifies the print services attribute identifier.

attrValue

(IN) Points to the attribute value.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

The type of the buffer pointed to by *attrValue* depends on the value of *attrID*. If *attrID* is not recognized, an error is returned.

See **NWPSCfgAddPrintQueueAttr** for a list of legal attributes and corresponding types.

NCP Calls

None

See Also

NWPSCfgAddPrintQueueAttr, **NWPSCfgGetFirstPrintQueueAttr**, **NWPSCfgGetNextPrintQueueAttr**, **NWPSCfgEndNextPrintQueueAttr**, **NWPSCfgGetFirstPrintQueue**, **NWPSCfgGetNextPrintQueue**, **NWPSCfgEndNextPrintQueue**, **NWPSCfgVerifyPrintQueue**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSCfgDeletePrintServer

Deletes a print server from the Bindery or the Directory
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgDeletePrintServer (
   WORD          connType,
   DWORD         connID,
   char NWFAR  *pServerName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

pServerName

(IN) Points to the name of the print server to delete.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

If *connType* is NWPS_BINDERY_SERVICE, **NWPSCfgDeletePrintServer** also deletes all of the print server's printers, but leaves all queues intact. All configuration information is also removed from the Bindery.

If *connType* is NWPS_DIRECTORY_SERVICE, **NWPSCfgDeletePrintServer** removes all attributes of the print server from the current context, but leaves printers and queues intact.

NCP Calls

None

See Also

NWPSCfgAddPrintServer, **NWPSCfgDeletePrintServerAttr**, **NWPSCfgGetFirstPrintServer**, **NWPSCfgGetNextPrintServer**, **NWPSCfgEndNextPrintServer**, **NWPSCfgVerifyPrintServer**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSCfgDeletePrintServerAttr

Deletes a print server attribute from a print server object
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgDeletePrintServerAttr (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *pServerName,
   WORD          attrID,
   void NWFAR   *attrValue);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

pServerName

(IN) Points to the name of the print server.

attrID

(IN) Specifies the print server attribute identifier.

attrValue

(IN) Points to the attribute value.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

If an attempt to delete a required attribute is made, an error is returned.

The type of the buffer pointed to by *attrValue* depends on the value of *attrID*. If *attrID* is not recognized, an error is returned.

If the attribute is not required and is single-valued, the attribute is removed. If the attribute is not required and is multi-valued, the value specified is deleted, and the other values are left in place.

See **NWPSCfgAddPrintServerAttr** for a list of legal attributes and corresponding types.

NCP Calls

None

See Also

NWPSCfgAddPrintServerAttr, **NWPSCfgDeletePrintServer**, **NWPSCfgGetFirstPrintServerAttr**, **NWPSCfgGetNextPrintServerAttr**, **NWPSCfgEndNextPrintServerAttr**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSCfgEndNextNServer

Frees up memory and performs other cleanups associated with the scan started by **NWPSCfgGetFirstNServer**

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgEndNextNServer (
   NWPSListHandle   handle);
```

Parameters

handle

(IN) Specifies the value returned from **NWPSCfgGetFirstNServer**.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery Error

Remarks

NWPSCfgEndNextNServer is required only if the handle returned from **NWPSCfgGetFirstNServer** is non-zero. Calling **NWPSCfgEndNextNServer** with a handle of zero has no effect.

NCP Calls

None

See Also

NWPSCfgGetFirstNServer, **NWPSCfgGetNextNServer**, **NWPSCfgVerifyNServer**

NWPSCfgEndNextPrinter

Frees up memory and performs other cleanups associated with the scan started by **NWPSCfgGetFirstPrinter**

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgEndNextPrinter (
   NWPSListHandle   handle);
```

Parameters

handle

(IN) Specifies the value returned from **NWPSCfgGetFirstPrinter**.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

NWPSCfgEndNextPrinter is required only if the handle returned from **NWPSCfgGetFirstPrinter** is non-zero. Calling **NWPSCfgEndNextPrinter** with a handle of zero has no effect.

NCP Calls

None

See Also

NWPSCfgGetFirstPrinter, **NWPSCfgGetNextPrinter**, **NWPSCfgVerifyPrinter**

NWPSCfgEndNextPrinterAttr

Frees up memory and performs other cleanups associated with the scan started by **NWPSCfgGetFirstPrinterAttr**

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgEndNextPrinterAttr (
   NWPSListHandle   handle);
```

Parameters

handle

(IN) Specifies the value returned from **NWPSCfgGetFirstPrinterAttr**.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

NWPSCfgEndNextPrinterAttr is required only if the handle returned from **NWPSCfgGetFirstPrinterAttr** is non-zero. Calling **NWPSCfgEndNextPrinterAttr** with a handle of zero has no effect.

NCP Calls

None

See Also

NWPSCfgGetFirstPrinterAttr, **NWPSCfgGetNextPrinterAttr**

NWPSCfgEndNextPrintQueue

Frees up memory and performs other cleanups associated with the scan started by **NWPSCfgGetFirstPrintQueue**

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgEndNextPrintQueue (
   NWPSListHandle   handle);
```

Parameters

handle

(IN) Specifies the value returned from **NWPSCfgGetFirstPrintQueue**.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

NWPSCfgEndNextPrintQueue is required only if the handle returned from **NWPSCfgGetFirstPrintQueue** is non-zero. Calling **NWPSCfgEndNextPrintQueue** with a handle of zero has no effect.

NCP Calls

None

See Also

NWPSCfgGetFirstPrintQueue, **NWPSCfgGetNextPrintQueue**, **NWPSCfgVerifyPrintQueue**

NWPSCfgEndNextPrintQueueAttr

Frees up memory and performs other cleanups associated with the scan started by **NWPSCfgGetFirstPrintQueueAttr**

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgEndNextPrintQueueAttr (
   NWPSListHandle   handle);
```

Parameters

handle

(IN) Specifies the value returned from **NWPSCfgGetFirstPrintQueueAttr**.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

NWPSCfgEndNextPrintQueueAttr is required only if the handle returned from **NWPSCfgGetFirstPrintQueueAttr** is non-zero. Calling **NWPSCfgEndNextPrintQueueAttr** with a handle of zero has no effect.

NCP Calls

None

See Also

NWPSCfgGetFirstPrintQueueAttr, **NWPSCfgGetNextPrintQueueAttr**

NWPSCfgEndNextPrintServer

Frees up memory and performs other cleanups associated with the scan started by **NWPSCfgGetFirstPrintServer**

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrvr.h

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgEndNextPrintServer (
   NWPSListHandle   handle);
```

Parameters

handle

(IN) Specifies the value returned from **NWPSCfgGetFirstPrintServer**.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

NWPSCfgEndNextPrintServer is required only if the handle returned from **NWPSCfgGetFirstPrintServer** is non-zero. Calling **NWPSCfgEndNextPrintServer** with a handle of zero has no effect.

NCP Calls

None

See Also

NWPSCfgGetFirstPrintServer, **NWPSCfgGetNextPrintServer**, **NWPSCfgVerifyPrintServer**

NWPSCfgEndNextPrintServerAttr

Frees up memory and performs other cleanups associated with the scan started by **NWPSCfgGetFirstPrintServerAttr**

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgEndNextPrintServerAttr (
   NWPSListHandle   handle);
```

Parameters

handle

(IN) Specifies the value returned from **NWPSCfgGetFirstPrintServerAttr**.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

NWPSCfgEndNextPrintServerAttr is required only if the handle returned from **NWPSCfgGetFirstPrintServerAttr** is non-zero. Calling **NWPSCfgEndNextPrintServerAttr** with a handle of zero has no effect.

NCP Calls

None

See Also

NWPSCfgGetFirstPrintServerAttr, **NWPSCfgGetNextPrintServerAttr**

NWPSCfgGetFirstNServer

Finds the first NetWare server in the list serviced by the print server
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgGetFirstNServer (
   WORD                   connType,
   DWORD                  connID,
   char NWFAR            *pServerName,
   NWPSListHandle NWFAR  *handle,
   char NWFAR            *nServerName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE or NWPS_BINDERY_SERVICE_PRE_40.

connID

(IN) Specifies the connection identifier.

pServerName

(IN) Points to the name of the print server.

handle

(OUT) Points to the value to be passed to **NWPSCfgGetNextNServer** and **NWPSCfgEndNextNServer**.

nServerName

(OUT) Points to the name of the first NetWare server found. Its maximum length is 48 characters including NULL termination.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

Memory allocated and files opened by **NWPSCfgGetFirstNServer** must be respectively deallocated and closed by calling **NWPSCfgEndNextNServer**. If the completion code is non-zero, the value returned for the handle is zero, and calling **NWPSCfgEndNextNServer** is not required.

NCP Calls

None

See Also

NWPSCfgEndNextNServer, **NWPSCfgGetNextNServer**, **NWPSCfgVerifyNServer**

NWPSCfgGetFirstPrinter

Finds the first printer in the list serviced by the print server
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgGetFirstPrinter
  (WORD                    connType,
   DWORD                   connID,
   NWPSListHandle NWFAR   *handle,
   char NWFAR             *pServerName,
   char NWFAR            *printerName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

handle

(OUT) Points to the value to be passed to **NWPSCfgGetNextPrinter** and **NWPSCfgEndNextPrinter**.

pserverName

(IN/OUT) Points to the name of the print server. Its maximum length is MAX_DN_BYTES for Directory Services; 48 characters for other services.

printerName

(OUT) Points to the name of the first printer found. Its maximum length is MAX_DN_BYTES for Directory Services; 48 characters for other services.

Return Values

0x0000	Successful

0x7760	NWPSE_END_OF_LIST
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

Memory allocated and files opened by **NWPSCfgGetFirstPrinter** must be respectively deallocated and closed by **NWPSCfgEndNextPrinter**. If the completion code is non-zero, the value returned for the handle is zero, and calling **NWPSCfgEndNextPrinter** is not required.

pserverName is optional in NWPS_DIRECTORY_SERVICE mode; it is required in NWPS_BINDERY_SERVICE mode. If present, only the printers for that particular print server are scanned.

In Directory Services, if *pserverName* points to an empty string, all printers are scanned; and their associated print server names are also returned. If *pserverName* is NULL, all printers are scanned but no print server names are returned. The latter method is the fastest for Directory Services.

NCP Calls

None

See Also

NWPSCfgEndNextPrinter, **NWPSCfgGetNextPrinter**, **NWPSCfgVerifyPrinter**

NWPSCfgGetFirstPrinterAttr

Finds the first printer attribute value
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgGetFirstPrinterAttr (
   WORD                   connType,
   DWORD                  connID,
   char NWFAR            *pServerName,
   char NWFAR            *printerName,
   WORD                   attrID,
   NWPSListHandle NWFAR  *handle,
   void NWFAR           *attrValue);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

pServerName

(IN) Points to the name of the print server (optional in NWPS_DIRECTORY_SERVICE).

printerName

(IN) Points to the name of the printer.

attrID

(IN) Specifies the print services attribute identifier.

handle

(OUT) Points to the value to be passed to **NWPSCfgGetNextPrinterAttr** and **NWPSCfgEndNextPrinterAttr** (optional).

attrValue

(OUT) Points to the buffer in which to place the first attribute found.

Its maximum length is MAX_DN_BYTES + sizeof(NWPSTypedName).

Return Values

0x0000	Successful
0x7772	NWPSE_ERROR_EXPANDING_DB
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

Memory allocated and files opened by **NWPSCfgGetFirstPrinterAttr** must be respectively deallocated and closed by calling **NWPSCfgEndNextPrinterAttr**. If the completion code is non-zero, or if the attribute requested is single-valued, the value returned for the handle is zero. In this case, calling **NWPSCfgEndNextPrinterAttr** is not required.

The type of the buffer pointed to by *attrValue* depends on the value of *attrID.* If *attrID* is not recognized, an error is returned.

pServerName is required for calls where *connType* is NWPS_BINDERY_SERVICE; it is ignored when *connType* is NWPS_DIRECTORY_SERVICE. Bindery printers do not exist independent of their print server.

handle is optional. If *handle* is NULL, **NWPSCfgEndNextPrinterAttr** is called by **NWPSCfgGetFirstPrinterAttr** to release any allocated resources. Normally calls to **NWPSCfgGetFirstPrinterAttr** for single-valued attributes should pass a NULL pointer to the handle. Calls to **NWPSCfgGetFirstPrinterAttr** for multi-valued attributes may also pass a NULL pointer if only the first value in the list is desired, or if the caller is trying only to determine the existence of elements in the list.

Legal attribute identifiers and values for the printer configuration follow:

Bindery Identifier	**Directory Identifier**	**Type**	**Multi Valued**
None	NWPS_ATTR_ACL	Object_ACL_T	Yes
None	NWPS_ATTR_CART	char[ ]	Yes
NWPS_ATTR_CN	NWPS_ATTR_CN	char[ ]	No

NWPS_ATTR_CONF	NWPS_ATTR_CONF	Octet_String_T	No
NWPS_ATTR_DESC	NWPS_ATTR_DESC	char[ ]	No
NWPS_ATTR_DQUEUE	NWPS_ATTR_DQUEUE	char[ ]	No
None	NWPS_ATTR_HOST_DEV	char[ ]	No
None	NWPS_ATTR_MEMORY	Integer_T	No
None	NWPS_ATTR_NADD	Net_Address_T	Yes
None	NWPS_ATTR_NADD_REST	Net_Address_T	Yes
NWPS_ATTR_NOTIFY	NWPS_ATTR_NOTIFY	NWPS_Typed_Name	Yes
NWPS_ATTR_OPER	NWPS_ATTR_OPER	NWPS_Typed_Name	Yes
NWPS_ATTR_OWNER	NWPS_ATTR_OWNER	char[ ]	Yes
None	NWPS_ATTR_PAGE	char[ ]	Yes
NWPS_ATTR_PRINT_SER	NWPS_ATTR_PRINT_SER	Typed_Name_T (The printer number is obtained from typed name's level field.)	No
NWPS_ATTR_QUEUE	NWPS_ATTR_QUEUE	Typed_Name_T	Yes
None	NWPS_ATTR_SEE_ALSO	char[ ]	Yes
None	NWPS_ATTR_SERIAL	char[ ]	Yes
NWPS_ATTR_STAT	NWPS_ATTR_STAT	Integer_T	No
None	NWPS_ATTR_TYPE (faces)	char[ ]	Yes

NOTE: char[ ] is a NULL-terminated ASCII string.

NWPS_ATTR_STAT values follow:

```
0 = NWPS_PSTAT_JOB_WAIT
1 = NWPS_PSTAT_FORM_WAIT
2 = NWPS_PSTAT_PRINTING
3 = NWPS_PSTAT_PAUSED
4 = NWPS_PSTAT_STOPPED
5 = NWPS_PSTAT_MARK_EJECT
6 = NWPS_PSTAT_READY_TO_DOWN
7 = NWPS_PSTAT_NOT_CONNECTED
8 = NWPS_PSTAT_PRIVATE
```

NCP Calls

None

See Also

NWPSCfgEndNextPrinterAttr, NWPSCfgGetNextPrinterAttr

NWPSCfgGetFirstPrintQueue

Finds the first print queue on a NetWare server or in a directory context
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgGetFirstPrintQueue (
   WORD                    connType,
   DWORD                   connID,
   NWPSListHandle NWFAR   *handle,
   char NWFAR            *queueName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

handle

(OUT) Points to the value passed by **NWPSCfgGetNextPrintQueue** and **NWPSCfgEndNextPrintQueue**.

queueName

(OUT) Points to the name of the first print queue found. Its maximum length is MAX_DN_BYTES for Directory Services; 48 characters for other services.

Return Values

0x0000	Successful
0x7760	NWPSE_END_OF_LIST
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

Files opened and memory allocated by **NWPSCfgGetFirstPrintQueue** must be respectively closed and deallocated by calling **NWPSCfgEndNextPrintQueue**. If the completion code is non-zero, the value returned for the handle is zero, and calling **NWPSCfgEndNextPrintQueue** is not required.

NCP Calls

None

See Also

NWPSCfgEndNextPrintQueue, **NWPSCfgGetNextPrintQueue**, **NWPSCfgVerifyPrintQueue**

NWPSCfgGetFirstPrintQueueAttr

Finds the first attribute for the print queue and returns all the values for the specified attribute

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgGetFirstPrintQueueAttr
  (WORD                   connType,
   DWORD                  connID,
   char NWFAR            *queueName,
   WORD                   attrID,
   NWPSListHandle NWFAR   handle,
   void NWFAR            *attrValue);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

queueName

(IN) Points to the name of the queue.

attrID

(IN) Specifies the print services attribute identifier.

handle

(OUT) Points to the value to be passed to **NWPSCfgGetNextPrintQueueAttr** and **NWPSCfgEndNextPrintQueueAttr** (optional).

attrValue

(OUT) Points to the buffer in which to place the first attribute found. Its maximum length is MAX_DN_BYTES + sizeof(NWPSTypedName).

Return Values

0x0000	Successful
0x7762	NWPSE_END_OF_ATTR_LIST
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

NWPSCfgGetFirstPrintQueueAttr allocates memory and opens files that must be respectively deallocated and closed by calling **NWPSCfgEndNextPrintQueueAttr**. If the completion code is non-zero or if the attribute requested is single-valued, the value returned for the handle is zero. In this case, calling **NWPSCfgEndNextPrintQueueAttr** is not required.

The type of the buffer pointed to by *attrValue* depends on the value of *attrID*. If *attrID* is not recognized, an error is returned.

handle is optional. If it is NULL, **NWPSCfgEndNextPrintQueueAttr** is called by **NWPSCfgGetFirstPrintQueueAttr** to release any allocated resources. Normally calls to **NWPSCfgGetFirstPrintQueueAttr** for single-valued attributes should pass a NULL pointer to the handle. Calls to **NWPSCfgGetFirstPrintQueueAttr** for multi-valued attributes may also pass a NULL pointer if only the first value in the list is desired, or if the caller is trying only to determine the existence of elements in the list.

Legal attribute identifiers for the printer's queue follow:

Bindery Identifier	Directory Identifier	Type	Multi Valued
None	NWPS_ATTR_ACL	Object_ACL_T	Yes
NWPS_ATTR_CN	NWPS_ATTR_CN	char[ ]	No
NWPS_ATTR_DESC	NWPS_ATTR_DESC	char[ ]	No
NWPS_ATTR_DEVICE	NWPS_ATTR_DEVICE	char[ ]	Yes
NWPS_ATTR_HOST_RES	NWPS_ATTR_HOST_RES	char[ ]	No
NWPS_ATTR_HOST_SER	NWPS_ATTR_HOST_SER	char[ ]	No
None	NWPS_ATTR_NA	Net_Address_	Yes

	DD	T	
NWPS_ATTR_OPER	NWPS_ATTR_OPER	NWPS_Typed_Name	Yes
NWPS_ATTR_QUE_DIR	NWPS_ATTR_QUE_DIR	NWPS_Typed_Name	Yes
None	NWPS_ATTR_SEE_ALSO	char[ ]	Yes
NWPS_ATTR_SERVER	NWPS_ATTR_SERVER	char[ ]	Yes
NWPS_ATTR_USER	NWPS_ATTR_USER	NWPS_Typed_Name	Yes
NWPS_ATTR_VOLUME	NWPS_ATTR_VOLUME	char[ ]	No

NOTE: char[ ] is a NULL-terminated ASCII string.

NCP Calls

None

See Also

NWPSCfgEndNextPrintQueueAttr, **NWPSCfgGetNextPrintQueueAttr**

NWPSCfgGetFirstPrintServer

Finds the first print server in the Bindery or Directory
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgGetFirstPrintServer
  (WORD                    connType,
   DWORD                   connID,
   NWPSListHandle NWFAR   *handle,
   char NWFAR            *pServerName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

handle

(OUT) Points to the value to be passed to **NWPSCfgGetNextPrintServer** and **NWPSCfgEndNextPrintServer**.

pServerName

(OUT) Points to the name of the first print server found. Its maximum length is MAX_DN_BYTES for Directory Services; 48 characters for other services.

Return Values

0x0000	Successful
0x7760	NWPSE_END_OF_LIST
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

Memory allocated and files opened by **NWPSCfgGetFirstPrintServer** must be respectively deallocated and closed by calling **NWPSCfgEndNextPrintServer**. If the completion code is non-zero, the value returned for the handle is zero, calling **NWPSCfgEndNextPrintQueueAttr**is not required.

NCP Calls

None

See Also

NWPSCfgEndNextPrintServer, **NWPSCfgGetNextPrintServer**, **NWPSCfgVerifyPrintServer**

NWPSCfgGetFirstPrintServerAttr

Retrieves the first print server attribute value for the indicated print server
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgGetFirstPrintServerAttr (
   WORD                   connType,
   DWORD                  connID,
   char NWFAR            *pServerName,
   WORD                   attrID,
   NWPSListHandle NWFAR  *handle,
   void NWFAR            *attrValue);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

pServerName

(IN) Points to the name of the print server.

attrID

(IN) Specifies the print services attribute identifier.

handle

(OUT) Points to the value to be passed to **NWPSCfgGetNextPrintServerAttr** and **NWPSCfgEndNextPrintServerAttr** (optional).

attrValue

(OUT) Points to the buffer in which to place the first attribute found. Its maximum length is MAX_DN_BYTES + sizeof(NWPSTypedName).

Return Values

0x0000	Successful
0x7760	NWPSE_END_OF_ATTR_LIST
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

Memory allocated and files opened by **NWPSCfgGetFirstPrintServerAttr** must be respectively deallocated and closed by calling **NWPSCfgEndNextPrintServerAttr**. If the completion code is non-zero or if the attribute requested is single-valued, the value returned for the handle is zero. In this case, calling **NWPSCfgEndNextPrintServerAttr** is not required.

The type of the buffer pointed to by *attrValue* depends on the value of *attrID*. If *attrIDs* is not recognized, an error is returned.

Legal attribute identifier and values for the print server are as follows:

Bindery Identifier	Directory Identifier	Type	Multi Valued
None	NWPS_ATTR_ACL	Object_ACL_T	Yes
NWPS_ATTR_CN	NWPS_ATTR_CN	char[ ]	No
NWPS_ATTR_DESC	NWPS_ATTR_DESC	char[ ]	No
None	NWPS_ATTR_HOST_DEV	char[ ]	No
None	NWPS_ATTR_NADD	Net_Address_T	Yes
NWPS_ATTR_OPER	NWPS_ATTR_OPER	NWPS_Typed_Name	Yes
NWPS_ATTR_PRINTER	NWPS_ATTR_PRINTER	Typed_Name_T	Yes
None	NWPS_ATTR_PRIV_KEY	Octet_String_T	No
None	NWPS_ATTR_PUBL_KEY	Octet_String_T	No
NWPS_ATTR_DEVICE	NWPS_ATTR_SAP	char[ ]	No

None	NWPS_ATTR_SEE_ALSO	char[ ]	Yes
None	NWPS_ATTR_STAT	Integer_T	No
NWPS_ATTR_USER	NWPS_ATTR_USER	char[ ]	Yes
None	NWPS_ATTR_VERS	char[ ]	No

NOTE: char[ ] is a NULL-terminated ASCII string.

NWPS_ATTR_STAT values follow:

```
0 = NWPS_RUNNING
1 = NWPS_GOING_DOWN
2 = NWPS_DOWN
3 = NWPS_INITIALIZING
```

NCP Calls

None

See Also

NWPSCfgEndNextPrintServerAttr, **NWPSCfgGetNextPrintServerAttr**

NWPSCfgGetNextNServer

Retrieves the next NetWare server name from the list of NetWare servers based on the information contained in the handle

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgGetNextNServer (
   NWPSListHandle   handle,
   char NWFAR      *nServerName);
```

Parameters

handle

(IN) Specifies the value returned from **NWPSCfgGetFirstNServer**.

nServerName

(OUT) Points to the name of the next NetWare server found. Its maximum length is 48 bytes.

Return Values

0x0000	Successful
0x7762	NWPSE_END_OF_LIST
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

If the handle is zero, NWPSE_END_OF_LIST is returned, indicating the end of the list of NetWare servers.

NCP Calls

None

See Also

NWPSCfgGetFirstNServer, NWPSCfgEndNextNServer, NWPSCfgVerifyNServer

NWPSCfgGetNextPrinter

Retrieves the next printer name from the list, based on the information contained in the handle

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgGetNextPrinter (
   NWPSListHandle   handle,
   char NWFAR      *pServerName,
   char NWFAR      *printerName);
```

Parameters

handle

(IN) Specifies the value returned from **NWPSCfgGetFirstPrinter**.

pServerName

(OUT) Points to the name of the print server (optional).

printerName

(OUT) Points to the name of the next printer found. Its maximum length is MAX_DN_BYTES for Directory Services; 48 characters for other services.

Return Values

0x0000	Successful
0x7760	NWPSE_END_OF_LIST
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

If the handle is zero, NWPSE_END_OF_LIST is returned, indicating the end of the list of printers.

pServerName must match the parameter passed in

NWPSCfgGetFirstPrinter. If a print server name or a NULL was passed in **NWPSCfgGetFirstPrinter**, *pServerName* is ignored. However, if the address of an empty string was passed to **NWPSCfgGetFirstPrinter**, data for all printers includes print servers; and the address of a destination buffer for the print server should be passed to **NWPSCfgGetFirstPrinter**.

NOTE: A NULL pointer is allowed for *pServerName* in this case, but it is highly inefficient because the data is retrieved from the NetWare server only to be discarded by **NWPSCfgGetFirstPrinter**.

NCP Calls

None

See Also

NWPSCfgGetFirstPrinter, **NWPSCfgEndNextPrinter**, **NWPSCfgVerifyPrinter**

NWPSCfgGetNextPrinterAttr

Retrieves the next printer attribute from the list, based on the information contained in the handle

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgGetNextPrinterAttr (
   NWPSListHandle   handle,
   char NWFAR      *printerName,
   void NWFAR      *attrValue);
```

Parameters

handle

(IN) Specifies the value returned from **NWPSCfgGetFirstPrinterAttr**.

printerName

(IN) Points to the name of the printer.

attrValue

(OUT) Points to the buffer in which to place the next attribute found. Its maximum length is MAX_DN_BYTES + sizeof(NWPSTypedName).

Return Values

0x0000	Successful
0x7760	NWPSE_END_OF_ATTR_LIST
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

If the handle is zero, NWPSE_END_OF_ATTR_LIST is returned, indicating the end of the list of printer attributes.

printerName is optional when getting attributes for the Bindery. In this case, the passed value should be NULL.

NCP Calls

None

See Also

NWPSCfgGetFirstPrinterAttr, **NWPSCfgEndNextPrinterAttr**

NWPSCfgGetNextPrintQueue

Retrieves the next print queue name from the list, based on the information contained in the handle

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgGetNextPrintQueue (
   NWPSListHandle   handle,
   char NWFAR      *queueName);
```

Parameters

handle

(IN) Specifies the value returned from **NWPSCfgGetFirstPrintQueue**.

queueName

(OUT) Points to the name of the next print queue found. Its maximum length is MAX_DN_BYTES for Directory Services; 48 characters for other services.

Return Values

0x0000	Successful
0x7762	NWPSE_END_OF_LIST
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

If the handle is zero, NWPSE_END_OF_LIST is returned, indicating the end of the list of print queues.

NCP Calls

None

See Also

NWPSCfgGetFirstPrintQueue, **NWPSCfgEndNextPrintQueue**, **NWPSCfgVerifyPrintQueue**

NWPSCfgGetNextPrintQueueAttr

Retrieves the next print queue attribute from the list, based on the information contained in the handle

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgGetNextPrintQueueAttr (
   NWPSListHandle   handle,
   char NWFAR      *queueName,
   void NWFAR      *attrValue);
```

Parameters

handle

(IN) Specifies the value returned from **NWPSCfgGetFirstPrintQueueAttr**.

queueName

(IN) Points to the name of the queue.

attrValue

(OUT) Points to the buffer in which to place the next attribute found. Its maximum length is MAX_DN_BYTES + sizeof(NWPSTypedName).

Return Values

0x0000	Successful
0x7762	NWPSE_END_OF_ATTR_LIST
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

If the handle is zero, NWPSE_END_OF_ATTR_LIST is returned,

indicating the end of the list of print queue attributes.

NCP Calls

None

See Also

NWPSCfgGetFirstPrintQueueAttr, **NWPSCfgEndNextPrintQueueAttr**

NWPSCfgGetNextPrintServer

Retrieves the next print server name from the list, based on the information contained in the handle

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE)NWPSCfgGetNextPrintServer (
   NWPSlistHandle   handle,
   char NWFAR      *pServerName);
```

Parameters

handle

(IN) Specifies the value returned from **NWPSCfgGetFirstPrintServer**.

pServerName

(OUT) Points to the name of the next print server found. Its maximum length is MAX_DN_BYTES for Directory Services; 48 characters for other services.

Return Values

0x0000	Successful
0x7760	NWPSE_END_OF_LIST
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

If the handle is zero, NWPSE_END_OF_LIST is returned, indicating the end of the list of print servers.

NCP Calls

None

See Also

NWPSCfgGetFirstPrintServer, **NWPSCfgEndNextPrintServer**, **NWPSCfgVerifyPrintServer**

NWPSCfgGetNextPrintServerAttr

Retrieves the next print server attribute from the list, based on the information contained in the handle

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgGetNextPrintServerAttr (
   NWPSListHandle   handle,
   char NWFAR      *pServerName,
   void NWFAR      *attrValue);
```

Parameters

handle

(IN) Specifies the value returned from **NWPSCfgGetFirstPrintServerAttr**.

pServerName

(IN) Points to the name of the print server.

attrValue

(OUT) Points to the buffer in which to place the next attribute found. Its maximum length is MAX_DN_BYTES + sizeof(NWPSTypedName).

Return Values

0x0000	Successful
0x7762	NWPSE_END_OF_ATTR_LIST
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

If the handle is zero, NWPSE_END_OF_ATTR_LIST is returned,

indicating the end of the list of print server attributes.

NCP Calls

None

See Also

NWPSCfgGetFirstPrintServerAttr, **NWPSCfgEndNextPrintServerAttr**

NWPSCfgGetPrinterDefaults

Returns default printer settings for the specified type of printer
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgGetPrinterDefaults (
   WORD                  printerType,
   WORD                  subtype,
   NWPS_PConfig NWFAR  *pConfig);
```

Parameters

printerType

(IN) Specifies the type of printer.

subtype

(IN) Specifies the port number for the printer.

pConfig

(OUT) Points to NWPS_PConfig containing default settings.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

If *printerType* is set to NWPS_DEFAULT, the default printer type is selected.

If *subtype* is set to NWPS_DEFAULT, the default port is selected.*subtype* indicates the port number as follows:

```
NWPS_PORT_1        0
NWPS_PORT_2        1
NWPS_PORT_3        2
NWPS_PORT_4        3
NWPS_PORT_5        4
NWPS_PORT_6        5
NWPS_PORT_7        6
NWPS_PORT_8        7
NWPS_PORT_9        8
NWPS_PORT_10       9
```

NCP Calls

None

See Also

NWPSCfgAddPrinterAttr, **NWPSCfgGetFirstPrinterAttr**, **NWPSCfgGetNextPrinterAttr**, **NWPSCfgEndNextPrinterAttr**

NWPSCfgModifyPrinterAttr

Changes a printer attribute value in the Bindery or Directory
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgModifyPrinterAttr (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *pServerName,
   char NWFAR   *printerName,
   WORD          attrID,
   void NWFAR   *oldValue,
   void NWFAR   *newValue);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

pServerName

(IN) Points to the name of the print server.

printerName

(IN) Points to the name of the printer.

attrID

(IN) Specifies the print services attribute identifier.

oldValue

(IN) Points to the attribute value to remove.

newValue

(IN) Points to the attribute value to add.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

NWPSCfgModifyPrinterAttr fails if the old value cannot be matched exactly or if the new value is invalid.

The only way to change single-valued, required attributes is by calling **NWPSCfgModifyPrinterAttr**.

The type of the buffer pointed to by *oldValue* and *newValue* depends on the value of *attrID*. If *attrID* is not recognized, an error is returned.

Legal attribute identifiers and values for the printer configuration follow:

Bindery Identifier	Directory Identifier	Type	Multi Valued
None	NWPS_ATTR_ACL	Object_ACL_T	Yes
None	NWPS_ATTR_CART	char[ ]	Yes
NWPS_ATTR_CN	NWPS_ATTR_CN	char[ ]	No
NWPS_ATTR_CONF	NWPS_ATTR_CONF	Octet_String_T	No
NWPS_ATTR_DESC	NWPS_ATTR_DESC	char[ ]	No
NWPS_ATTR_DQUEUE	NWPS_ATTR_DQUEUE	char[ ]	No
None	NWPS_ATTR_HOST_DEV	char[ ]	No
None	NWPS_ATTR_MEMORY	Integer_T	No
None	NWPS_ATTR_NADD	Net_Address_T	Yes
None	NWPS_ATTR_NADD_REST	Net_Address_T	Yes
NWPS_ATTR_NOTIFY	NWPS_ATTR_NOTIFY	NWPS_Typed_Name	Yes
NWPS_ATTR_OPER	NWPS_ATTR_OPER	NWPS_Typed_Name	Yes

NWPS_ATTR_OWNER	NWPS_ATTR_OWNER	char[ ]	Yes
None	NWPS_ATTR_PAGE	char[ ]	Yes
NWPS_ATTR_PRINT_SER	NWPS_ATTR_PRINT_SER	Typed_Name_T	No
NWPS_ATTR_QUEUE	NWPS_ATTR_QUEUE	Typed_Name_T	Yes
None	NWPS_ATTR_SEE_ALSO	char[ ]	Yes
None	NWPS_ATTR_SERIAL	char[ ]	Yes
NWPS_ATTR_STAT	NWPS_ATTR_STAT	Integer_T	No
None	NWPS_ATTR_TYPE (faces)	char[ ]	Yes

NOTE: char[ ] is a NULL-terminated ASCII string.

NWPS_ATTR_STAT values follow:

```
0 = NWPS_PSTAT_JOB_WAIT
1 = NWPS_PSTAT_FORM_WAIT
2 = NWPS_PSTAT_PRINTING
3 = NWPS_PSTAT_PAUSED
4 = NWPS_PSTAT_STOPPED
5 = NWPS_PSTAT_MARK_EJECT
6 = NWPS_PSTAT_READY_TO_DOWN
7 = NWPS_PSTAT_NOT_CONNECTED
8 = NWPS_PSTAT_PRIVATE
```

NCP Calls

None

See Also

NWPSCfgAddPrinterAttr, **NWPSCfgDeletePrinterAttr**, **NWPSCfgGetFirstPrintServer**, **NWPSCfgGetNextPrintServer**, **NWPSCfgEndNextPrintServer**, **NWPSCfgVerifyPrintServer**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSCfgModifyPrintQueueAttr

Changes the old print queue attribute value to a new value
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgModifyPrintQueueAttr (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *queueName,
   WORD          attrID,
   void NWFAR   *oldValue,
   void NWFAR   *newValue);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

queueName

(IN) Points to the queue name to modify.

attrID

(IN) Specifies the print services attribute identifier.

oldValue

(IN) Points to the old attribute value.

newValue

(IN) Points to the new attribute value.

Return Values

0x0000	Successful
-1	General Error

other values	Bindery or Directory Services Errors

Remarks

NWPSCfgModifyPrintQueueAttr fails if the old value cannot be matched exactly or if the new value is invalid.

The only way to change single-valued, required attributes is by calling **NWPSCfgModifyPrintQueueAttr**.

The type of the buffer pointed to by *oldValue* and *newValue* depends on the value of *attrID*. If *attrID* is not recognized, an error is returned.

Legal attribute identifiers for the printer's queue follow:

Bindery Identifier	Directory Identifier	Type	Multi Valued
None	NWPS_ATTR_ACL	Object_ACL_T	Yes
NWPS_ATTR_CN	NWPS_ATTR_CN	char[ ]	No
NWPS_ATTR_DESC	NWPS_ATTR_DESC	char[ ]	No
NWPS_ATTR_DEVICE	NWPS_ATTR_DEVICE	char[ ]	Yes
NWPS_ATTR_HOST_RES	NWPS_ATTR_HOST_RES	char[ ]	No
NWPS_ATTR_HOST_SER	NWPS_ATTR_HOST_SER	char[ ]	No
None	NWPS_ATTR_NADD	Net_Address_T	Yes
NWPS_ATTR_OPER	NWPS_ATTR_OPER	NWPS_Typed_Name	Yes
NWPS_ATTR_QUE_DIR	NWPS_ATTR_QUE_DIR	NWPS_Typed_Name	Yes
None	NWPS_ATTR_SEE_ALSO	char[ ]	Yes
NWPS_ATTR_SERVER	NWPS_ATTR_SERVER	char[ ]	Yes
NWPS_ATTR_USER	NWPS_ATTR_USER	NWPS_Typed_Name	Yes
NWPS_ATTR_VOLUME	NWPS_ATTR_VOLUME	char[ ]	No

NOTE: char[ ] is a NULL-terminated ASCII string.

NCP Calls

None

See Also

NWPSCfgAddPrintQueueAttr, **NWPSCfgDeletePrintQueueAttr**, **NWPSCfgGetFirstPrintQueueAttr**, **NWPSCfgGetNextPrintQueueAttr**, **NWPSCfgEndNextPrintQueueAttr**, **NWPSCfgGetFirstPrintQueue**, **NWPSCfgGetNextPrintQueue**, **NWPSCfgEndNextPrintQueue**, **NWPSCfgVerifyPrintQueue**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSCfgModifyPrintServerAttr

Changes a print server attribute from one value to another in the Bindery or Directory

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgModifyPrintServerAttr (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *pServerName,
   WORD          attrID,
   void NWFAR   *oldValue,
   void NWFAR   *newValue);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

pServerName

(IN) Points to the name of the print server.

attrID

(IN) Specifies the print server attribute identifier.

oldValue

(IN) Points to the old attribute value.

newValue

(IN) Points to the new attribute value.

Return Values

0x0000	Successful

-1	General Error
other values	Bindery or Directory Services Errors

Remarks

NWPSCfgModifyPrintServerAttr fails if the old value cannot be matched exactly or if the new value is invalid.

The only way to change single-valued, required attributes is by calling **NWPSCfgModifyPrintServerAttr**.

The type of the buffer pointed to by *oldValue* and *newValue* depends on the value of *attrID*. If *attrID* is not recognized, an error is returned.

Legal attribute identifier and values for the print server are as follows:

Bindery Identifier	Directory Identifier	Type	Multi Valued
None	NWPS_ATTR_ACL	Object_ACL_T	Yes
NWPS_ATTR_CN	NWPS_ATTR_CN	char[ ]	No
NWPS_ATTR_DESC	NWPS_ATTR_DESC	char[ ]	No
None	NWPS_ATTR_HOST_DEV	char[ ]	No
None	NWPS_ATTR_NADD	Net_Address_T	Yes
NWPS_ATTR_OPER	NWPS_ATTR_OPER	NWPS_Typed_Name	Yes
NWPS_ATTR_PRINTER	NWPS_ATTR_PRINTER	Typed_Name_T	Yes
None	NWPS_ATTR_PRIV_KEY	Octet_String_T	No
None	NWPS_ATTR_PUBL_KEY	Octet_String_T	No
NWPS_ATTR_DEVICE	NWPS_ATTR_SAP	char[ ]	No
None	NWPS_ATTR_SEE_ALSO	char[ ]	Yes
None	NWPS_ATTR_STAT	Integer_T	No
NWPS_ATTR_USER	NWPS_ATTR_USER	char[ ]	Yes

ER			
None	NWPS_ATTR_VERS	char[ ]	No

NOTE: char[ ] is a NULL-terminated ASCII string.

NWPS_ATTR_STAT values follow:

```
0 = NWPS_RUNNING
1 = NWPS_GOING_DOWN
2 = NWPS_DOWN
3 = NWPS_INITIALIZING
```

NCP Calls

None

See Also

NWPSCfgAddPrintServerAttr, **NWPSCfgDeletePrintServerAttr**, **NWPSCfgGetFirstPrintServerAttr**, **NWPSCfgGetNextPrintServerAttr**, **NWPSCfgEndNextPrintServerAttr**, **NWPSCfgGetFirstPrintServer**, **NWPSCfgGetNextPrintServer**, **NWPSCfgEndNextPrintServer**, **NWPSCfgVerifyPrintServer**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSCfgVerifyNServer

Searches for and verifies the server name in the list of NetWare servers serviced by the indicated print server

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgVerifyNServer (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *pServerName,
   char NWFAR   *nServerName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE or NWPS_BINDERY_SERVICE_PRE_40.

connID

(IN) Specifies the connection identifier.

pServerName

(IN) Points to the name of the print server.

nServerName

(IN) Points to the name of the NetWare server to verify.

Return Values

0x0000	Successful
0x7761	NWPSE_NO_SUCH_LIST_ENTRY
-1	General Error
other values	Bindery or Directory Services error

NCP Calls

None

See Also

NWPSCfgGetFirstNServer, **NWPSCfgGetNextNServer**, **NWPSCfgEndNextNServer**

NWPSCfgScanNServer (obsolete 6/96)

Finds the list of NetWare servers serviced by a specified print server but is now obsolete. Call **NWPSCfgGetFirstNServer**, **NWPSCfgGetNextNServer**, **NWPSCfgEndNextNServer**, or **NWPSCfgVerifyNServer** instead

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

NWCCODE NWAPI NWPSCfgScanNServer
  (WORD           connType,
   DWORD          connID,
   DWORD NWFAR   *sequence,
   char NWFAR    *pServerName,
   char NWFAR    *nServerName);
```

Parameters

connType

(IN) Specifies the connection type (NWPS_BINDERY_SERVICE only).

connID

(IN) Specifies the connection identifier.

sequence

(IN/OUT) Points to the index. Set *sequence* to -1 before the first call; it is incremented by **NWPSCfgScanNServer (obsolete 6/96)** on every return.

pServerName

(IN) Points to the name of the print server.

nServerName

(IN/OUT) Points to the name of the NetWare server found. Its maximum length is 48 bytes.

Return Values

0x0000	Successful
-1	General Error

other values	Bindery or Directory Services Errors

Remarks

NWPSCfgScanNServer (obsolete 6/96) is valid only when *connType* is NWPS_BINDERY_SERVICE.

To find all the NetWare servers, set *sequence* to -1 and call **NWPSCfgScanNServer (obsolete 6/96)** repeatedly until an error is returned.

To verify a specific NetWare server is in a list of serviced NetWare servers, set *sequence* to NULL and pass the name of the NetWare server in *nServerName*.

NCP Calls

None

See Also

NWPSCfgAddNServer, **NWPSCfgDeleteNServer**, **NWPSCfgGetFirstNServer**, **NWPSCfgGetNextNServer**, **NWPSCfgEndNextNServer**, **NWPSCfgVerifyNServer**, **NWScanObject**, **NWCCScanConnRefs**, **NWCCOpenConnByRef**, **NWDSCreateContextHandle**

NWPSCfgScanPrinter (obsolete 6/96)

Finds a printer from the list of printers serviced by a print server but is now obsolete. Call **NWPSCfgGetFirstPrinter**, **NWPSCfgGetNextPrinter**, **NWPSCfgEndNextPrinter**, or **NWPSCfgVerifyPrinter** instead.

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

NWCCODE NWAPI NWPSCfgScanPrinter
  (WORD           connType,
   DWORD          connID,
   DWORD NWFAR   *sequence,
   char NWFAR    *pServerName,
   char NWFAR    *printerName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

sequence

(IN/OUT) Points to the index. Set *sequence* to -1 before the first call; it is incremented by **NWPSCfgScanPrinter (obsolete 6/96)** on every return.

pServerName

(IN/OUT) Points to the name of the print server. Its maximum length is MAX_DN_BYTES.

printerName

(IN/OUT) Points to the name of the printer. Its maximum length is MAX_DN_BYTES.

Return Values

0x0000	Successful

-1	General Error
other values	Bindery or Directory Services Errors

Remarks

To find all printers, set *sequence* to -1 before the first call. **NWPSCfgScanPrinter (obsolete 6/96)** updates and returns a new sequence number with every call.

To verify the existence of a printer, set *sequence* to NULL and set *printerName* to the name of the printer to verify.

pserverName is optional in NWPS_DIRECTORY_SERVICE mode; it is required in NWPS_BINDERY_SERVICE mode. If present, only the printers for that particular print server are scanned.

In Directory Services, if *pserverName* points to an empty string, all printers are scanned, and their associated print server names are also returned. If *pserverName* is NULL, all printers are scanned, but no print server names are returned. The latter method is the fastest for Directory Services.

NCP Calls

None

See Also

NWPSCfgAddPrinterAttr, **NWPSCfgDeletePrinter**, **NWPSCfgAddPrinter**, **NWPSCfgDeletePrinterAttr**, **NWPSCfgGetFirstPrinter**, **NWPSCfgGetNextPrinter**, **NWPSCfgEndNextPrinter**, **NWPSCfgVerifyPrinter**, **NWCCScanConnRefs**, **NWCCOpenConnByRef**, **NWDSCreateContextHandle**

NWPSCfgScanPrinterAttr (obsolete 6/96)

Finds all the values of a printer attribute but is now obsolete. Call **NWPSCfgGetFirstPrinterAttr**, **NWPSCfgGetNextPrinterAttr**, or **NWPSCfgEndNextPrinterAttr** instead

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

NWCCODE NWAPI NWPSCfgScanPrinterAttr
  (WORD           connType,
   DWORD          connID,
   DWORD NWFAR   *sequence,
   char NWFAR    *pServerName,
   char NWFAR    *printerName,
   WORD           attrID,
   void NWFAR    *attrValue);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

sequence

(IN/OUT) Points to the index. Set *sequence* to -1 before the first call; it is incremented by **NWPSCfgScanPrinterAttr (obsolete 6/96)** on every return.

pServerName

(IN) Points to the name of the print server.

printerName

(IN) Points to the name of the printer.

attrID

(IN) Specifies the print services attribute identifier.

attrValue

(OUT) Points to the attribute buffer. Its maximum length is MAX_DN_BYTES + sizeof(NWPSTypedName).

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

The type of the buffer pointed to by *attrValue* depends on the value of *attrID*. If *attrID* is not recognized, an error is returned.

attrValue must point to a valid buffer space for the requested attribute. For example, a string attribute must have a pointer to an array of bytes. An octet string must have a pointer to Octet_String_T with a valid length and pointer to an array of bytes.

Legal attribute identifiers and values for the printer configuration follow:

Bindery Identifier	**Directory Identifier**	**Type**	**Multi Valued**
None	NWPS_ATTR_ACL	Object_ACL_T	Yes
None	NWPS_ATTR_CART	char[ ]	Yes
NWPS_ATTR_CN	NWPS_ATTR_CN	char[ ]	No
NWPS_ATTR_CONF	NWPS_ATTR_CONF	Octet_String_T	No
NWPS_ATTR_DESC	NWPS_ATTR_DESC	char[ ]	No
NWPS_ATTR_DQUEUE	NWPS_ATTR_DQUEUE	char[ ]	No
None	NWPS_ATTR_HOST_DEV	char[ ]	No
None	NWPS_ATTR_MEMORY	Integer_T	No
None	NWPS_ATTR_NADD	Net_Address_T	Yes
None	NWPS_ATTR_NADD_REST	Net_Address_T	Yes

NWPS_ATTR_NOTIFY	NWPS_ATTR_NOTIFY	NWPS_Typed_Name	Yes
NWPS_ATTR_OPER	NWPS_ATTR_OPER	char[ ]	Yes
NWPS_ATTR_OWNER	NWPS_ATTR_OWNER	char[ ]	Yes
None	NWPS_ATTR_PAGE	char[ ]	Yes
NWPS_ATTR_PRINT_SER	NWPS_ATTR_PRINT_SER	Typed_Name_T	No
NWPS_ATTR_QUEUE	NWPS_ATTR_QUEUE	Typed_Name_T	Yes
None	NWPS_ATTR_SEE_ALSO	char[ ]	Yes
None	NWPS_ATTR_SERIAL	char[ ]	Yes
NWPS_ATTR_STAT	NWPS_ATTR_STAT	Integer_T	No
None	NWPS_ATTR_TYPE(faces)	char[ ]	Yes

NOTE: char[ ] is a NULL-terminated ASCII string.

NWPS_ATTR_STAT values follow:

```
0 = NWPS_PSTAT_JOB_WAIT
1 = NWPS_PSTAT_FORM_WAIT
2 = NWPS_PSTAT_PRINTING
3 = NWPS_PSTAT_PAUSED
4 = NWPS_PSTAT_STOPPED
5 = NWPS_PSTAT_MARK_EJECT
6 = NWPS_PSTAT_READY_TO_DOWN
7 = NWPS_PSTAT_NOT_CONNECTED
8 = NWPS_PSTAT_PRIVATE
```

To find the first value of a multi-valued attribute, set *sequence* to -1 before calling **NWPSCfgScanPrinterAttr (obsolete 6/96)**. *sequence* is updated internally in preparation for each call. *attrValue* should be a buffer large enough to hold each attribute value.

NCP Calls

None

See Also

NWPSCfgAddPrinterAttr, **NWPSCfgDeletePrinterAttr**, **NWPSCfgModifyPrinterAttr**, **NWPSCfgGetFirstPrinterAttr**,

NWPSCfgGetNextPrinterAttr, **NWPSCfgEndNextPrinterAttr**, **NWCCScanConnRefs**, **NWCCOpenConnByRef**, **NWDSCreateContextHandle**

NWPSCfgScanPrintQueue (obsolete 6/96)

Finds all the print queues on a NetWare server or in a Directory context but is now obsolete. Call **NWPSCfgGetFirstPrintQueue**, **NWPSCfgGetNextPrintQueue**, **NWPSCfgEndNextPrintQueue**, or **NWPSCfgVerifyPrintQueue** instead.

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

NWCCODE NWAPI NWPSCfgScanPrintQueue
   (WORD           connType,
    DWORD          connID,
    DWORD NWFAR   *sequence,
    char NWFAR    *queueName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

sequence

(IN/OUT) Points to the index. Set *sequence* to -1 before the first call; it is incremented by **NWPSCfgScanPrintQueue (obsolete 6/96)** on every return.

queueName

(IN/OUT) Points to the name of the print queue for which to search. Its maximum length is MAX_DN_BYTES.

Return Values

0x0000	Successful
-1	General Error
other	Bindery or Directory Services Errors

values	

Remarks

To get the list, set *sequence* to -1 and make the call repeatedly until an error code is returned.

To verify that a queue name is valid, set *sequence* to NULL and specify the queue name in *queueName*.

NCP Calls

None

See Also

NWPSCfgAddPrintQueue, **NWPSCfgDeletePrintQueue**, **NWPSCfgGetFirstPrintQueueAttr**, **NWPSCfgGetNextPrintQueueAttr**, **NWPSCfgEndNextPrintQueueAttr**, **NWPSCfgVerifyPrintQueue**, **NWCCScanConnRefs**, **NWCCOpenConnByRef**, **NWDSCreateContextHandle**

NWPSCfgScanPrintQueueAttr (obsolete 6/96)

Finds an attribute for the print queue and returns all the values of the specified attribute but is now obsolete. Call **NWPSCfgGetFirstPrintQueueAttr**, **NWPSCfgGetNextPrintQueueAttr**, or **NWPSCfgEndNextPrintQueueAttr** instead.

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

NWCCODE NWAPI NWPSCfgScanPrintQueueAttr
   (WORD           connType,
    DWORD          connID,
    DWORD NWFAR   *sequence,
    char NWFAR    *queueName,
    WORD           attrID,
    void NWFAR    *attrValue);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

sequence

(IN/OUT) Points to the index. Set *sequence* to -1 before the first call; it is incremented by **NWPSCfgScanPrintQueueAttr (obsolete 6/96)** on every return.

queueName

(IN) Points to the queue name to scan.

attrID

(IN) Specifies the print services attribute identifier.

attrValue

(OUT) Points to the attribute value.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

The type of the buffer pointed to by *attrValue* depends on the value of *attrID*. If *attrID* is not recognized, an error is returned.

Set *sequence* to -1 before calling **NWPSCfgScanPrintQueueAttr (obsolete 6/96)** the first time.

Legal attribute identifiers for the printer's queue follow:

Bindery Identifier	Directory Identifier	Type	Multi Valued
None	NWPS_ATTR_ACL	Object_ACL_T	Yes
NWPS_ATTR_CN	NWPS_ATTR_CN	char[ ]	No
NWPS_ATTR_DESC	NWPS_ATTR_DESC	char[ ]	No
NWPS_ATTR_DEVICE	NWPS_ATTR_DEVICE	char[ ]	Yes
NWPS_ATTR_HOST_RES	NWPS_ATTR_HOST_RES	char[ ]	No
NWPS_ATTR_HOST_SER	NWPS_ATTR_HOST_SER	char[ ]	No
None	NWPS_ATTR_NADD	Net_Address_T	Yes
NWPS_ATTR_OPER	NWPS_ATTR_OPER	char[ ]	Yes
NWPS_ATTR_QUE_DIR	NWPS_ATTR_QUE_DIR	NWPS_Typed_Name	Yes
None	NWPS_ATTR_SEE_ALSO	char[ ]	Yes
NWPS_ATTR_SERVER	NWPS_ATTR_SERVER	char[ ]	Yes
NWPS_ATTR_USER	NWPS_ATTR_USER	NWPS_Typed_Name	Yes

NWPS_ATTR_VOLUME	NWPS_ATTR_VOLUME	char[ ]	No

NOTE: char[ ] is a NULL-terminated character string.

NCP Calls

None

See Also

NWPSCfgAddPrintQueueAttr, **NWPSCfgDeletePrintQueueAttr**, **NWPSCfgModifyPrintQueueAttr**, **NWPSCfgGetFirstPrintQueueAttr**, **NWPSCfgGetNextPrintQueueAttr**, **NWPSCfgEndNextPrintQueueAttr**, **NWCCScanConnRefs**, **NWCCOpenConnByRef**, **NWDSCreateContextHandle**

NWPSCfgScanPrintServer (obsolete 6/96)

Finds a print server in the bindery or directory but is now obsolete. Call **NWPSCfgGetFirstPrintServer**, **NWPSCfgGetNextPrintServer**, **NWPSCfgEndNextPrintServer**, or **NWPSCfgVerifyPrintServer** instead.

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

NWCCODE NWAPI NWPSCfgScanPrintServer
  (WORD           connType,
   DWORD          connID,
   DWORD NWFAR   *sequence,
   char NWFAR    *pServerName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

sequence

(IN/OUT) Points to the index. Set *sequence* to -1 before the first call; it is incremented by **NWPSCfgScanPrintServer (obsolete 6/96)** on every return.

pServerName

(IN/OUT) Points to the name of the print server to scan. Its maximum length is MAX_DN_BYTES.

Return Values

0x0000	Successful
0x89FC	NO_SUCH_OBJECT (shell error)
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

If *sequence* is -1, **NWPSCfgScanPrintServer (obsolete 6/96)** returns the first print server it finds.

If *sequence* is NULL, the name of the print server in *pServerName* is verified.

NCP Calls

None

See Also

NWPSCfgAddPrintServer, **NWPSCfgDeletePrintServer**, **NWPSCfgGetFirstPrintServer**, **NWPSCfgGetNextPrintServer**, **NWPSCfgEndNextPrintServer**, **NWPSCfgVerifyPrintServer**, **NWCCScanConnRefs**, **NWCCOpenConnByRef**, **NWDSCreateContextHandle**

NWPSCfgScanPrintServerAttr (obsolete 6/96)

Finds the value of a given print server attribute but is now obsolete. Call **NWPSCfgGetFirstPrintServerAttr**, **NWPSCfgGetNextPrintServerAttr**, or **NWPSCfgEndNextPrintServerAttr** instead

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

NWCCODE NWAPI NWPSCfgScanPrintServerAttr
  (WORD           connType,
   DWORD          connID,
   DWORD NWFAR   *sequence,
   char NWFAR    *pServerName,
   WORD           attrID,
   void NWFAR    *attrValue);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

sequence

(IN/OUT) Points to the index. Set *sequence* to -1 before the first call; it is incremented by **NWPSCfgScanPrintServerAttr (obsolete 6/96)** on every return.

pServerName

(IN) Points to the name of the print server.

attrID

(IN) Specifies the print server attribute identifier.

attrValue

(OUT) Points to the attribute value. Its maximum length is MAX_DN_BYTES + sizeof(NWPSTypedName).

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

The type of the buffer pointed to by *attrValue* depends on the value of *attrID*. If *attrID* is not recognized, an error is returned.

An illegal attribute name causes an error to be returned.

Legal attribute identifier and values for the print server are as follows:

Bindery Identifier	**Directory Identifier**	**Type**	**Multi Valued**
None	NWPS_ATTR_ACL	Object_ACL_T	Yes
NWPS_ATTR_CN	NWPS_ATTR_CN	char[ ]	No
NWPS_ATTR_DESC	NWPS_ATTR_DESC	char[ ]	No
None	NWPS_ATTR_HOST_DEV	char[ ]	No
None	NWPS_ATTR_NADD	Net_Address_T	Yes
NWPS_ATTR_OPER	NWPS_ATTR_OPER	char[ ]	Yes
NWPS_ATTR_PRINTER	NWPS_ATTR_PRINTER	Typed_Name_T	Yes
None	NWPS_ATTR_PRIV_KEY	Octet_String_T	No
None	NWPS_ATTR_PUBL_KEY	Octet_String_T	No
NWPS_ATTR_DEVICE	NWPS_ATTR_SAP	char[ ]	No
None	NWPS_ATTR_SEE_ALSO	char[ ]	Yes
None	NWPS_ATTR_STAT	Integer_T	No
NWPS_ATTR_USER	NWPS_ATTR_USER	char[ ]	Yes

None	NWPS_ATTR_VERS	char[ ]	No

NOTE: char[ ] is a NULL-terminated ASCII string.

NWPS_ATTR_STAT values follow:

```
0 = NWPS_RUNNING
1 = NWPS_GOING_DOWN
2 = NWPS_DOWN
3 = NWPS_INITIALIZING
```

On the first call, *sequence* should be set to -1, *attrID* is set to identify the attribute to read, and *attrValue* is a pointer to the buffer to write the attribute value to.

NCP Calls

None

See Also

NWPSCfgAddPrintServerAttr, **NWPSCfgDeletePrintServerAttr**, **NWPSCfgGetFirstPrintServerAttr**, **NWPSCfgGetNextPrintServerAttr**, **NWPSCfgEndNextPrintServerAttr**, **NWPSCfgModifyPrintServerAttr**, **NWCCScanConnRefs**, **NWCCOpenConnByRef**, **NWDSCreateContextHandle**

NWPSCfgVerifyPrinter

Searches for and verifies the printer name in the list of printers serviced by the indicated print server

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgVerifyPrinter (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *pServerName,
   char NWFAR   *printerName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

pServerName

(IN) Points to the name of the print server.

printerName

(IN) Points to the name of the printer to verify.

Return Values

0x0000	Successful
0x7761	NWPSE_NO_SUCH_LIST_ENTRY
0x7763	NWPSE_WRONG_CLASS_LIST_ENTRY
-1	General Error
other values	Bindery or Directory Services Errors

NCP Calls

None

See Also

NWPSCfgGetFirstPrinter, **NWPSCfgGetNextPrinter**, **NWPSCfgEndNextPrinter**

NWPSCfgVerifyPrintQueue

Searches for and verifies the name of the queue on the list of print queues on a NetWare server or in a Directory context

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgVerifyPrintQueue (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *queueName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

queueName

(IN) Points to the name of the print queue to verify.

Return Values

0x0000	Successful
0x7761	NWPSE_NO_SUCH_LIST_ENTRY
0x7763	NWPSE_WRONG_CLASS_LIST_ENTRY (Directory Services only)
-1	General Error
other values	Bindery or Directory Services Errors

NCP Calls

None

See Also

NWPSCfgGetFirstPrintQueue, **NWPSCfgGetNextPrintQueue**, **NWPSCfgEndNextPrintQueue**

NWPSCfgVerifyPrintServer

Searches for and verifies the print server name in the list of print servers on a NetWare server or in a Directory context

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSCfgVerifyPrintServer (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *pServerName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

pServerName

(IN) Points to the name of the print server to verify.

Return Values

0x0000	Successful
0x7761	NWPSE_NO_SUCH_LIST_ENTRY
0x7763	NWPSE_WRONG_CLASS_LIST_ENTRY (Directory Services only)
-1	General Error
other values	Bindery or Directory Services Errors

NCP Calls

None

See Also

NWPSCfgGetFirstPrintServer, **NWPSCfgGetNextPrintServer**, **NWPSCfgEndNextPrintServer**

NWPSComAbortPrintJob

Aborts the current print job and clears the buffer
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComAbortPrintJob (
   NWPS_SPXID_HANDLE   spxID,
   nuint               printerID,
   nuint               jobOutcome);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

printerID

(IN) Specifies the printer (from 0 to the maximum number of printers minus 1).

jobOutcome

(IN) Specifies what to do with the print job after it is aborted.

Return Values

0x0000	NWPSE_SUCCESSFUL
0x00ED	NWPSE_PSC_CONNECTION_TERMIN
0x0301	NWPSE_NOT_ENOUGH_MEMORY
0x0302	NWPSE_NO_SUCH_PRINTER
0x0303	NWPSE_INVALID_PARAMETER
0x0307	NWPSE_NOT_CONNECTED
0x0306	NWPSE_GOING_DOWN
0x0309	NWPSE_NO_JOB_ACTIVE
0x030C	NWPSE_DOWN
0x030E	NWPSE_NO_RIGHTS

Remarks

NWPSComAbortPrintJob ejects one blank page unless suppress form feed is set. It is for operators and users who have attached and logged in to the print server.

jobOutcome values are as follows:

```
1 = NWPS_RETURN_TO_QUEUE
2 = NWPS_THROW_AWAY
```

NCP Calls

None

See Also

NWPSComAttachToPrintServer, **NWPSComLoginToPrintServer**, **NWPSComStopPrinter**

NWPSComAddNotifyObject

Adds an object to the print server's list of objects notified when a printer needs attention (such as mounting forms, printer off-line, or out of paper)

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComAddNotifyObject (
   NWPS_SPXID_HANDLE   spxID,
   nuint               printerID,
   pnstr8              nServerName,
   pnstr8              objectName,
   nuint               objectType,
   nuint               notifyDelay,
   nuint               notifyInterval);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

printerID

(IN) Specifies the printer (0 to maximum number of printers minus 1).

nServerName

(IN) Points to the name of the NetWare server.

objectName

(IN) Points to the name of object to be notified.

objectType

(IN) Specifies the type of object to be notified.

notifyDelay

(IN) Specifies the time before first notice.

notifyInterval

(IN) Specifies the time between notices.

Return Values

These are common return values; see 'Return Values" in the *NetWare SDK*

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL
0x00ED	NWPSE_CONNECTION_TERMINATED
0x0103	NWPSE_BINDERY_LOCKED
0x0301	NWPSE_NOT_ENOUGH_MEMORY
0x0302	NWPSE_NO_SUCH_PRINTER
0x030A	NWPSE_NOT_ATTACHED_TO_SERVER
0x030B	NWPSE_ALREADY_IN_LIST
0x030C	NWPSE_DOWN
0x030E	NWPSE_NO_RIGHTS

Remarks

The object is specified by *nServerName*, *objectName*, and *objectType*. OT_WILD indicates the job owner must be notified. The object must have an active connection ID to be notified.

Possible object type values are:

```
 1    =    OT_USER
 2    =    OT_USER_GROUP
-1    =    OT_WILD
```

NWPSComAddNotifyObject is for operators only.

NCP Calls

None

See Also

NWPSComChangeNotifyInterval, **NWPSComDeleteNotifyObject**, **NWPSComGetNotifyObject**, **NWPSComAttachToPrintServer**, **NWPSComLoginToPrintServer**, **NWScanObject**

NWPSComAddQueueToPrinter

Assigns a queue to a printer and assigns a priority to the queue
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComAddQueueToPrinter (
   NWPS_SPXID_HANDLE   spxID,
   nuint               printerID,
   pnstr8              nServerName,
   pnstr8              queueName,
   nuint               priority);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

printerID

(IN) Specifies the printer (0 to maximum number of printers minus 1).

nServerName

(IN) Points to the name of the queue NetWare server.

queueName

(IN) Points to the name of the queue to be added.

priority

(IN) Specifies the priority of this queue.

Return Values

These are common return values; see 'Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL
0x00ED	NWPSE_CONNECTION_TERMINATED

0x0103	NWPSE_BINDERY_LOCKED
0x0201	NWPSE_NOT_AUTHORIZED_FOR_QUEUE
0x0202	NWPSE_QUEUE_HALTED
0x0203	NWPSE_UNABLE_TO_ATTACH_TO_QUEUE
0x0204	NWPSE_TOO_MANY_QUEUE_SERVERS
0x0301	NWPSE_NOT_ENOUGH_MEMORY
0x0302	NWPSE_NO_SUCH_PRINTER
0x030A	NWPSE_NOT_ATTACHED_TO_SERVER
0x030B	NWPSE_ALREADY_IN_LIST
0x030C	NWPSE_DOWN
0x030E	NWPSE_NO_RIGHTS

Remarks

The print server must be attached to the NetWare server where the queue is to be added. **NWPSComAddQueueToPrinter** attaches a queue server to the queue. The queue cannot have more than 25 print servers servicing it.

NWPSComAddQueueToPrinter is for operators only.

NCP Calls

None

See Also

NWPSComAttachPServerToNServer, **NWPSComAttachToPrintServer**, **NWPSComChangeQueuePriority**, **NWPSComDeleteQueueFromPrinter**, **NWPSComGetPrintersServicingQ**, **NWPSComGetQueuesServiced**, **NWPSComLoginToPrintServer**, **NWScanObject**

NWPSComAttachPServerToNServer

Attaches a print server to a NetWare server
NetWare Server: 2.2, 3.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComAttachPServerToNServer (
   NWPS_SPXID_HANDLE   spxID,
   pnstr8              nServerName,
   pnstr8               password);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

nServerName

(IN) Points to the name of the NetWare server to attach to.

password

(IN) Points to the print server's login password.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL
0x00ED	NWPSC_CONNECTION_TERMINATED
0x0101	NWPSE_TOO_MANY_NW_SERVERS
0x0102	NWPSE_UNKNOWN_NW_SERVER
0x0103	NWPSE_BINDERY_LOCKED
0x0	NWPSE_NW_SERVER_MAXED_OUT

104	
0x0 105	NWPSE_NO_RESPONSE
0x0 106	NWPSE_ALREADY_ATTACHED
0x0 107	NWPSE_CANT_ATTACH
0x0 108	NWPSE_NO_ACCOUNT_BALANCE
0x0 109	NWPSE_NO_CREDIT_LEFT
0x0 10A	NWPSE_INTRUDER_DETECTION_LOCK
0x0 10B	NWPSE_TOO_MANY_CONNECTIONS
0x0 10C	NWPSE_ACCOUNT_DISABLED
0x0 10D	NWPSE_UNAUTHORIZED_TIME
0x0 10E	NWPSE_UNAUTHORIZED_STATION
0x0 10F	NWPSE_NO_MORE_GRACE
0x0 110	NWPSE_LOGIN_DISABLED
0x0 111	NWPSE_ILLEGAL_ACCT_NAME
0x0 112	NWPSE_PASSWORD_HAS_EXPIRED
0x0 113	NWPSE_ACCESS_DENIED
0x0 114	NWPSE_CANT_LOGIN
0x0 301	NWPSE_NOT_ENOUGH_MEMORY
0x0 30C	NWPSE_DOWN
0x0 30E	NWPSE_NO_RIGHTS
0x0 30F	NWPSE_CMD_NOT_SUPPORTED

Remarks

Before calling **NWPSComAttachPServerToNServer**, a print server object with the same name must exist on the NetWare server.

If the print server attaches without an error, it also logs in to the NetWare server. If the login fails, the print server detaches from the NetWare server. If the attach and the login are successful, **NWPSComAttachPServerToNServer** reads the NetWare server's configuration files, which include the queues the print server will be servicing and the objects to notify.

NWPSComAttachPServerToNServer is for operators only.

NCP Calls

None

See Also

NWPSComAttachToPrintServer, **NWPSComDetachPServerFromNServer**, **NWPSComLoginToPrintServer**, **NWScanObject**

NWPSComAttachToPrintServer

Establishes an SPX connection between the workstation and the print server
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComAttachToPrintServer (
   nuint                     connType,
   nuint32                   connID,
   nuint                     timeout,
   pnstr8                    pserverName,
   NWPS_SPXID_HANDLE NWFAR  *spxID);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the NetWare server connection ID number or Directory Services context.

timeout

(IN) Specifies the number of seconds before time out.

pserverName

(IN) Points to the name of the print server to which to attach.

spxID

(OUT) Points to the number of theSPX™ connection to the print server.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0	NWPSE_SUCCESSFUL

000	
0x0 0ED	NWPSE_CONNECTION_TERMINATED
0x0 040	NWPSE_NO_AVAILABLE_SPX_ CONNECTION
0x0 041	NWPSE_SPX_NOT_INITIALIZED
0x0 042	NWPSE_NO_SUCH_SERVER
0x0 043	NWPSE_UNABLE_TO_GET_SERVER_ADDR
0x0 044	NWPSE_UNABLE_TO_CONNECT_TO SERV
0x0 045	NWPSE_NO_AVAILABLE_IPX_SOCKETS
0x0 046	NWPSE_ALREADY_ATTACHED_TO_A_ PRINT

Remarks

NWPSComAttachToPrintServer is for operators and users; call it before calling **NWPSComLoginToPrintServer**.

NCP Calls

None

See Also

NWPSComDetachFromPrintServer, **NWPSComLoginToPrintServer**, **NWScanObject**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSComCancelDownRequest

Cancels the **NWPSComDownPrintServer** command
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_com.h>
or
 #include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComCancelDownRequest (
   NWPS_SPXID_HANDLE   spxID);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0 000	NWPSE_SUCCESSFUL
0x0 0ED	NWPSE_CONNECTION_TERMINATED
0x0 30C	NWPSE_DOWN
0x0 30E	NWPSE_NO_RIGHTS

Remarks

NWPSComCancelDownRequest works only if the print server is processing a print job. If the print server has already advertised it is down, it returns NWPSE_DOWN.

NWPSComCancelDownRequest is for operators only.

NCP Calls

None

See Also

NWPSComAttachToPrintServer, **NWPSComLoginToPrintServer**, **NWPSComDownPrintServer**, **NWPSComGetPrintServerInfo**

NWPSComChangeNotifyInterval

Sets or changes the amount of time the print server waits before it notifies an object of a change, and the time interval the print server waits before sending succeeding notices

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComChangeNotifyInterval (
   NWPS_SPXID_HANDLE   spxID,
   nuint               printerID,
   pnstr8              nServerName,
   pnstr8              objectName,
   nuint               objectType,
   nuint               notifyDelay,
   nuint               notifyInterval);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

printerID

(IN) Specifies the printer (0 to maximum number of printers minus 1).

nServerName

(IN) Points to the name of the object's NetWare server.

objectName

(IN) Points to the name of the object to be notified.

objectType

(IN) Specifies the type of object to be notified.

notifyDelay

(IN) Specifies the number of seconds the print server waits before it notifies an object of a change.

notifyInterval

(IN) Specifies the number of seconds the print server waits between succeeding notices.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0 000	NWPSE_SUCCESSFUL
0x0 0ED	NWPSE_CONNECTION_TERMINATED
0x0 103	NWPSE_BINDERY_LOCKED
0x0 301	NWPSE_NOT_ENOUGH_MEMORY
0x0 302	NWPSE_NO_SUCH_PRINTER
0x0 30A	NWPSE_NOT_ATTACHED_TO_SERVER
0x0 30C	NWPSE_DOWN
0x0 30D	NWPSE_NOT_IN_LIST
0x0 30E	NWPSE_NO_RIGHTS

Remarks

The object is specified by *nServerName*, *objectName*, and *objectType*. A blank *objectName* or *objectType* OT_WILD indicates the notify interval is to be changed for the job owner. The object to notify must already be in the notify list.

Possible object type values are:

```
 1    =    OT_USER
 2    =    OT_USER_GROUP
-1    =    OT_WILD
```

NWPSComChangeNotifyInterval is for operators only.

NCP Calls

None

See Also

NWPSComAddNotifyObject, **NWPSComAttachToPrintServer**, **NWPSComDeleteNotifyObject**, **NWPSComGetNotifyObject**, **NWPSComLoginToPrintServer**, **NWScanObject**

NWPSComChangeQueuePriority

Changes the service priority assigned to the specified queue
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComChangeQueuePriority (
   NWPS_SPXID_HANDLE   spxID,
   nuint               printerID,
   pnstr8              nServerName,
   pnstr8              queueName,
   nuint                priority);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

printerID

(IN) Specifies the printer (0 to maximum number of printers minus 1).

nServerName

(IN) Points to the name of the object's NetWare server.

queueName

(IN) Points to the name of the queue to be changed.

priority

(IN) Specifies the new priority to give to the queue.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL
0x00ED	NWPSE_CONNECTION_TERMINATED
0x0103	NWPSE_BINDERY_LOCKED

0x0 301	NWPSE_NOT_ENOUGH_MEMORY
0x0 302	NWPSE_NO_SUCH_PRINTER
0x0 30A	NWPSE_NOT_ATTACHED_TO_SERVER
0x0 30C	NWPSE_DOWN
0x0 30D	NWPSE_NOT_IN_LIST
0x0 30E	NWPSE_NO_RIGHTS

Remarks

Before calling **NWPSComChangeQueuePriority**, the operator must be attached to the NetWare server where the queue exists. Attempts to get information about a queue will fail if the queue is not on the specified NetWare server or if the queue is not in the print server list.

NWPSComChangeQueuePriority is for operators only.

NCP Calls

None

See Also

NWPSComAddQueueToPrinter , **NWPSComAttachToPrintServer**, **NWPSComDeleteQueueFromPrinter**, **NWPSComGetPrintersServicingQ**, **NWPSComGetQueuesServiced**, **NWPSComLoginToPrintServer**, **NWScanObject**

NWPSComChangeServiceMode

Changes the queue service mode of the specified printer
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComChangeServiceMode (
   NWPS_SPXID_HANDLE   spxID,
   nuint               printerName,
   nuint               serviceMode);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

printerName

(IN) Specifies the printer (0 to maximum number of printers minus 1).

serviceMode

(IN) Specifies the new service mode; it must be a valid number (0-3).

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL
0x00ED	NWPSE_CONNECTION_TERMINATED
0x0103	NWPSE_BINDERY_LOCKED
0x0301	NWPSE_NOT_ENOUGH_MEMORY
0x0302	NWPSE_NO_SUCH_PRINTER
0x0	NWPSE_INVALID_PARAMETER

303	
0x0 30C	NWPSE_DOWN
0x0 30E	NWPSE_NO_RIGHTS

Remarks

serviceMode indicates whether forms should be changed as needed (NWPS_QUEUE_ONLY), form changes should be minimized within queues (NWPS_QUEUE_BEFORE_FORM), forms should never be changed (NWPS_FORM_ONLY), or form changes should be minimized across queues (NWPS_FORM_BEFORE_QUEUE). Valid numbers are listed below:

```
0 = NWPS_QUEUE_ONLY
1 = NWPS_QUEUE_BEFORE_FORM
2 = NWPS_FORM_ONLY
3 = NWPS_FORM_BEFORE_QUEUE
```

NWPSComChangeServiceMode is for operators only.

NCP Calls

None

See Also

NWPSComAttachToPrintServer, **NWPSComChangeQueuePriority**, **NWPSComLoginToPrintServer**

NWPSComDeleteNotifyObject

Deletes an object from the print server's list of objects notified when a printer needs attention (such as mounting forms, printer offline, or out of paper)

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComDeleteNotifyObject (
   NWPS_SPXID_HANDLE   spxID,
   nuint               printerID,
   pnstr8              nServerName,
   pnstr8              objectName,
   nuint               objectType);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

printerID

(IN) Specifies the printer (0 to maximum number of printers minus 1).

nServerName

(IN) Points to the name of the object's NetWare server.

objectName

(IN) Points to the name of the object to be notified.

objectType

(IN) Specifies the type of object to be notified.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL
0x0	NWPSE_CONNECTION_TERMINATED

0ED	
0x0 103	NWPSE_BINDERY_LOCKED
0x0 301	NWPSE_NOT_ENOUGH_MEMORY
0x0 302	NWPSE_NO_SUCH_PRINTER
0x0 30A	NWPSE_NOT_ATTACHED_TO_SERVER
0x0 30C	NWPSE_DOWN
0x0 30D	NWPSE_NOT_IN_LIST
0x0 30E	NWPSE_NO_RIGHTS

Remarks

The object is specified by *nServerName*, *objectName*, and *objectType*. A blank *objectName* or *objectType* OT_WILD indicates the job owner is to be deleted.

Possible object type values are:

```
 1    =    OT_USER
 2    =    OT_USER_GROUP
-1    =    OT_WILD
```

NWPSComDeleteNotifyObject is for operators only.

NCP Calls

None

See Also

NWPSComAddNotifyObject, **NWPSComAttachToPrintServer**, **NWPSComChangeNotifyInterval**, **NWPSComGetNotifyObject**, **NWPSComLoginToPrintServer**, **NWScanObject**

NWPSComDeleteQueueFromPrinter

Deletes a queue from a printer's service list
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComDeleteQueueFromPrinter (
   NWPS_SPXID_HANDLE   spxID,
   nuint               printerID,
   pnstr8              nServerName,
   pnstr8              queueName,
   nuint               detach,
   nuint               jobOutcome);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

printerID

(IN) Specifies the printer (0 to maximum number of printers minus 1).

nServerName

(IN) Points to the name of the object's NetWare server.

queueName

(IN) Points to the name of the queue to be deleted.

detach

(IN) Specifies whether to detach: 0=NO; 1=YES.

jobOutcome

(IN) Specifies what to do after the queue is deleted: 1 = NWPS_RETURN_TO_QUEUE; 2 = NWPS_THROW_AWAY. *jobOutcome* is used only if *detach* is YES.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0 000	NWPSE_SUCCESSFUL
0x0 0ED	NWPSE_CONNECTION_TERMINATED
0x0 103	NWPSE_BINDERY_LOCKED
0x0 301	NWPSE_NOT_ENOUGH_MEMORY
0x0 302	NWPSE_NO_SUCH_PRINTER
0x0 30A	NWPSE_NOT_ATTACHED_TO_SERVER
0x0 30C	NWPSE_DOWN
0x0 30D	NWPSE_NOT_IN_LIST
0x0 30E	NWPSE_NO_RIGHTS

Remarks

The server must be attached to the NetWare server where the queue exists. If a print server is servicing a job from the queue, **NWPSComDeleteQueueFromPrinter** aborts the job using *jobOutcome*.

NWPSComDeleteQueueFromPrinter is for operators only.

NCP Calls

None

See Also

NWPSComAddQueueToPrinter, **NWPSComAttachToPrintServer**, **NWPSComChangeQueuePriority**, **NWPSComDeleteQueueFromPrinter**, **NWPSComGetPrintersServicingQ**, **NWPSComGetQueuesServiced**, **NWPSComLoginToPrintServer**, **NWScanObject**

NWPSComDetachFromPrintServer

Disconnects the SPX™ connection between the workstation and the print server and logs the user out from the print server

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComDetachFromPrintServer (
   NWPS_SPXID_HANDLE   spxID);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL
0x00ED	NWPSE_CONNECTION_TERMINATED

Remarks

NWPSComDetachFromPrintServer is for operators and users.

NCP Calls

None

See Also

NWPSComAttachToPrintServer

NWPSComDetachPServerFromNServer

Detaches a print server from a NetWare server
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComDetachPServerFromNServer (
   NWPS_SPXID_HANDLE   spxID,
   pnstr8              nServerName,
   nuint               detach,
   nuint               jobOutcome);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

nServerName

(IN) Points to the name of the object's NetWare server.

detach

(IN) Specifies whether to detach: 0=NO; 1=YES.

jobOutcome

(IN) Specifies what to do after detaching: 1 = NWPS_RETURN_TO_QUEUE; 2 = NWPS_THROW_AWAY. *jobOutcome* is used only if *detach* is YES.

Return Values

These are common return values; see 'Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL
0x00ED	NWPSE_CONNECTION_TERMINATED
0x0103	NWPSE_BINDERY_LOCKED

0x0 301	NWPSE_NOT_ENOUGH_MEMORY
0x0 305	NWPSE_CANT_DETACH_PRIMARY_SERVER
0x0 30A	NWPSE_NOT_ATTACHED_TO_SERVER
0x0 30C	NWPSE_DOWN
0x0 30E	NWPSE_NO_RIGHTS

Remarks

If *detach* is set to YES and a job is being serviced, the print server acts according to the *jobOutcome* flags.

If *detach* is set to NO and a job is being serviced, the print server remains attached to the NetWare server even though **NWPSComDetachPServerFromNServer** has returned SUCCESSFUL, and detaches when the job has finished.

Before detaching, **NWPSComDetachPServerFromNServer** detaches all of the print server's printers from queues on the NetWare server. If any printers are servicing queues, it waits until they are finished before it detaches them. It also deletes the notify objects on the NetWare server after it detaches.

NWPSComDetachPServerFromNServer is for operators only.

NCP Calls

None

See Also

NWPSComAttachPServerToNServer, **NWPSComAttachToPrintServer**, **NWPSComLoginToPrintServer**, **NWScanObject**

NWPSComDownPrintServer

Shuts down the print server
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComDownPrintServer (
   NWPS_SPXID_HANDLE   spxID,
   nuint               immediate,
   nuint               jobOutcome);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

immediate

(IN) Specifies whether to shut down immediately: 0=NO; 1=YES.

jobOutcome

(IN) Specifies what to do after shut down: 1 = NWPS_RETURN_TO_QUEUE;2 = NWPS_THROW_AWAY. *jobOutcome* is used only if *immediate* is YES.

Return Values

These are common return values; see 'Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL
0x00ED	NWPSE_CONNECTION_TERMINATED
0x0301	NWPSE_NOT_ENOUGH_MEMORY
0x0303	NWPSE_INVALID_PARAMETER
0x0	NWPSE_DOWN

30C	
0x0 30E	NWPSE_NO_RIGHTS

Remarks

Before it shuts down, the print server checks the value of *jobOutcome* dealing with the job accordingly, stops the printer, and, depending on the value of *immediate*, advertises its status with GOING_DOWN or DOWN. *jobOutcome* must be a valid number.

NWPSComDownPrintServer does not terminate the SPX™ connection unless the print server goes down before it returns its completion code. The next call after the print server is downed returns NWPSE_CONNECTION_TERMINATED.

NWPSComDownPrintServer is for operators only.

NCP Calls

None

See Also

NWPSComAttachToPrintServer, **NWPSComCancelDownRequest**, **NWPSComLoginToPrintServer**

NWPSComEjectForm

Sends a form feed to the specified printer
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComEjectForm (
   NWPS_SPXID_HANDLE   spxID,
   nuint               printerID);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

printerID

(IN) Specifies the printer (from zero to the maximum number of printers minus 1).

Return Values

These are common return values; see 'Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL
0x00ED	NWPSE_CONNECTION_TERMINATED
0x0103	NWPSE_BINDERY_LOCKED
0x0301	NWPSE_NOT_ENOUGH_MEMORY
0x0302	NWPSE_NO_SUCH_PRINTER
0x0304	NWPSE_PRINTER_BUSY
0x0	NWPSE_NOT_CONNECTED

307	
0x0 30C	NWPSE_DOWN
0x0 30E	NWPSE_NO_RIGHTS

Remarks

NWPSComEjectForm is successful only when the printer is idle. If the printer is active, an error is returned.

NWPSComEjectForm is for operators only.

NCP Calls

None

See Also

NWPSComAttachPServerToNServer, **NWPSComGetPrinterStatus**, **NWPSComLoginToPrintServer**

NWPSComGetAttachedNServers

Returns the names of the NetWare servers attached to the print server
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComGetAttachedNServers (
   NWPS_SPXID_HANDLE   spxID,
   pnuint              sequence,
   pnstr8              nServerName);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

sequence

(IN) Points to the index. *sequence* must be set to 0 before the first call; it is incremented by **NWPSComGetAttachedNServers** on every call.

nServerName

(OUT) Points to the name of the NetWare server.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0 000	NWPSE_SUCCESSFUL
0x0 0ED	NWPSE_CONNECTION_TERMINATED
0x0 103	NWPSE_BINDERY_LOCKED
0x0 301	NWPSE_NOT_ENOUGH_MEMORY
0x0 30C	NWPSE_DOWN

0x3 0D	NWPSE_NOT_IN_LIST

Remarks

NWPSE_NOT_IN_LIST is returned when no NetWare servers are left in the list.

NWPSComGetAttachedNServers is for anyone.

NCP Calls

None

See Also

NWPSComAttachPServerToNServer, **NWPSComAttachToPrintServer**, **NWPSComLoginToPrintServer**

NWPSComGetExtPrinterStatus

Retrieves the status information about a printer
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComGetExtPrinterStatus (
   NWPS_SPXID_HANDLE   spxID,
   nuint               printerID,
   pnstr8              primaryStatus,
   pnuint              primaryLevel,
   pnstr8              secondaryStatus,
   pnuint              secondaryLevel,
   pnuint              activeJobCount,
   pnuint              serviceMode,
   pnuint              formsMounted,
   pnuint              formList);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

printerID

(IN) Specifies the printer (0 to maximum number of printers minus 1).

primaryStatus

(OUT) Points to the primary status.

primaryLevel

(OUT) Points to the primary error level.

secondaryStatus

(OUT) Points to the secondary status.

secondaryLevel

(OUT) Points to the secondary error level.

activeJobCount

(OUT) Points to number of active jobs.

serviceMode

(OUT) Points to the service mode.

formsMounted

(OUT) Points to the number of mounted forms.

formList

(OUT) Points to the array of the forms mounted (maximum of 20).

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0 000	NWPSE_SUCCESSFUL
0x0 0ED	NWPSE_CONNECTION_TERMINATED
0x0 103	NWPSE_BINDERY_LOCKED
0x0 301	NWPSE_NOT_ENOUGH_MEMORY
0x0 302	NWPSE_NO_SUCH_PRINTER
0x0 30C	NWPSE_DOWN
0x0 30E	NWPSE_NO_RIGHTS

Remarks

Before calling **NWPSComGetExtPrinterStatus**, the printer number must be known to the print server. Only status information for printers that can be accessed on the current SPX connection is returned. The client must disconnect and establish a new SPX connection to access printers on other print servers.

status values follow:

```
0 = NWPS_PSTAT_JOB_WAIT
1 = NWPS_PSTAT_FORM_WAIT
2 = NWPS_PSTAT_PRINTING
3 = NWPS_PSTAT_PAUSED
4 = NWPS_PSTAT_STOPPED
5 = NWPS_PSTAT_MARK_EJECT
6 = NWPS_PSTAT_READY_TO_DOWN
7 = NWPS_PSTAT_NOT_CONNECTED
8 = NWPS_PSTAT_PRIVATE
```

troubleCode values follow:

```
0 = NWPS_PRINTER_RUNNING
1 = NWPS_PRINTER_OFFLINE
2 = NWPS_PRINTER_PAPER_OUT
```

serviceMode values follow:

```
0 = NWPS_QUEUE_ONLY
1 = NWPS_QUEUE_BEFORE_FORM
2 = NWPS_FORM_ONLY
3 = NWPS FORM BEFORE QUEUE
```

NWPSComGetExtPrinterStatus is for operators and users.

NCP Calls

None

See Also

NWPSComAttachToPrintServer, **NWPSComLoginToPrintServer**

NWPSComGetNextRemotePrinter

Returns the printer number of the next available printer
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComGetNextRemotePrinter (
   NWPS_SPXID_HANDLE   spxID,
   pnuint              printerID,
   pnuint              printerType,
   pnstr8              printerName);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

printerID

(IN/OUT) Points to the printer number. Set *printerID* to -1 on the first call; thereafter it increments automatically. Printer numbers configured to active printers are skipped.

printerType

(OUT) Points to the type of printer.

printerName

(OUT) Points to the name of the printer.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL
0x00ED	NWPSE_CONNECTION_TERMINATED
0x0301	NWPSE_NOT_ENOUGH_MEMORY

0x0302	NWPSE_NO_SUCH_PRINTER
0x030C	NWPSE_DOWN

Remarks

The printer number that is returned by **NWPSComGetNextRemotePrinter** should be passed back on each subsequent call to get the next inactive printer number as returned in *printerID*.

Call **NWPSComGetNextRemotePrinter** to find all printers with the NOT CONNECTED status by setting the printer number to -1. Continue calling **NWPSComGetNextRemotePrinter** until 0x0302 is returned indicating that all available printers have been found.

printerType can have the following values:

Type	Value	Description
NWPS_P_PAR_1	0	Parallel port 1
NWPS_P_PAR_2	1	Parallel port 2
NWPS_P_PAR_3	2	Parallel port 3
NWPS_P_SER_1	3	Serial port 1
NWPS_P_SER_2	4	Serial port 2
NWPS_P_SER_3	5	Serial port 3
NWPS_P_SER_4	6	Serial port 4
NWPS_P_REM_PAR_1	7	NPrinter parallel port 1
NWPS_P_REM_PAR_2	8	NPrinter parallel port 2
NWPS_P_REM_PAR_3	9	NPrinter parallel port 3
NWPS_P_REM_SER_1	10	NPrinter serial port 1
NWPS_P_REM_SER_2	11	NPrinter serial port 2
NWPS_P_REM_SER_3	12	NPrinter serial port 3
NWPS_P_REM_SER_4	13	NPrinter serial port 4
NWPS_P_REM_	14	Other type of network printer

NWPS_P_REM_OTHER_1	14	Other type of network printer
NWPS_P_ELSEWHERE_1	15	Defined elsewhere
NWPS_P_XNP_1	16	Extended network printer
NWPS_P_LOC_AIO	17	AIO auto-start printer
NWPS_P_REM_AIO	18	AIO user-start printer
NWPS_P_APPLE_1	100	AppleTalk printer
NWPS_P_UNIX_1	200	UNIX user-start printer

NCP Calls

None

See Also

NWPSComAttachToPrintServer, **NWPSComGetPrinterStatus**, **NWPSComLoginToPrintServer**, **NWPSComRequestRemotePrinter**

NWPSComGetNotifyObject

Returns the object names and the notification intervals when a printer needs attention

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComGetNotifyObject (
   NWPS_SPXID_HANDLE   spxID,
   nuint               printerID,
   pnuint              sequence,
   pnstr8              nServerName,
   psntr8              objectName,
   pnuint              objectType,
   pnuint              notifyDelay,
   pnuint              notifyInterval);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

printerID

(IN) Specifies the printer (0 to maximum number of printers minus 1).

sequence

(IN/OUT) Points to the index. *sequence* must be set to 0 before the first call; it is incremented by **NWPSComGetNotifyObject** on every return.

nServerName

(OUT) Points to the name of the object's NetWare server.

objectName

(OUT) Points to the name of the object to be notified.

objectType

(OUT) Points to the type of object to be notified.

notifyDelay

(OUT) Points to the length of time before the first notice.

notifyInterval

(OUT) Points to the length of time between notices.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0 000	NWPSE_SUCCESSFUL
0x0 0ED	NWPSE_CONNECTION_TERMINATED
0x0 103	NWPSE_BINDERY_LOCKED
0x0 301	NWPSE_NOT_ENOUGH_MEMORY
0x0 302	NWPSE_NO_SUCH_PRINTER
0x0 30C	NWPSE_DOWN
0x0 30D	NWPSE_NOT_IN_LIST
0x0 30E	NWPSE_NO_RIGHTS

Remarks

NWPSComGetNotifyObject returns NWPSE_NOT_IN LIST when there are no objects left in the notify list.

NWPSComGetNotifyObject is for operators and users.

NCP Calls

None

See Also

NWPSComAddNotifyObject, **NWPSComAttachToPrintServer**, **NWPSComChangeNotifyInterval**, **NWPSComDeleteNotifyObject**, **NWPSComLoginToPrintServer**, **NWScanObject**

NWPSComGetPrintersServicingQ

Returns an array of bytes identifying which printers are servicing the specified queue

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE)NWPSComGetPrintersServicingQ (
   NWPS_SPXID_HANDLE   spxID,
   pnstr8              nServerName,
   pnstr8              queueName,
   nuint               maxPrinters,
   pnuint              actualPrinters,
   pnuint              printerArray);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

nServerName

(IN) Points to the name of the client's NetWare server.

queueName

(IN) Points to the name of the queue.

maxPrinters

(IN) Specifies the maximum number of printers to return.

actualPrinters

(OUT) Points to the actual number of printers returned.

printerArray

(OUT) Points to the array of bytes. A byte is allocated for each printer attached to the print server, up to the maximum number requested.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0 000	NWPSE_SUCCESSFUL
0x0 0ED	NWPSE_CONNECTION_TERMINATED
0x0 103	NWPSE_BINDERY_LOCKED
0x0 301	NWPSE_NOT_ENOUGH_MEMORY
0x0 30C	NWPSE_DOWN
0x0 30E	NWPSE_NO_RIGHTS

Remarks

The number of bytes in the array corresponds to the number of printers attached to the print server and each byte in the array represents a different printer. *maxPrinters* is the maximum number of printers to be returned.

NWPSComGetPrintersServicingQ is for operators and users.

NCP Calls

None

See Also

NWPSComAddQueueToPrinter, **NWPSComAttachToPrintServer**, **NWPSComDeleteQueueFromPrinter**, **NWPSComLoginToPrintServer**, **NWPSComGetQueuesServiced**

NWPSComGetPrinterStatus

Retrieves the status information about a printer
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComGetPrinterStatus (
   NWPS_SPXID_HANDLE   spxID,
   nuint               printerID,
   pnuint              status,
   pnuint              troubleCode,
   pnuint              active,
   pnuint              serviceMode,
   pnuint              formNumber,
   pnstr8              formName,
   pnstr8              printerName);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

printerID

(IN) Specifies the printer (0 to maximum number of printers minus 1).

status

(OUT) Points to the printer status.

troubleCode

(OUT) Points to the trouble code.

active

(OUT) Points to whether the printer is active: 0=No; 1=Yes.

serviceMode

(OUT) Points to the service mode.

formNumber

(OUT) Points to the number of the mounted form.

formName

(OUT) Points to the name of the form mounted from PRINTDEF.

printerName

(OUT) Points to the name of the printer.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0 000	NWPSE_SUCCESSFUL
0x0 0ED	NWPSE_CONNECTION_TERMINATED
0x0 103	NWPSE_BINDERY_LOCKED
0x0 301	NWPSE_NOT_ENOUGH_MEMORY
0x0 302	NWPSE_NO_SUCH_PRINTER
0x0 30C	NWPSE_DOWN
0x0 30E	NWPSE_NO_RIGHTS

Remarks

Before calling **NWPSComGetPrinterStatus**, the printer number must be known to the print server. Only status information for printers that can be accessed on the current SPX connection is returned. The client must disconnect and establish a new SPX connection to access printers on other print servers.

status values follow:

```
0 = NWPS_PSTAT_JOB_WAIT
1 = NWPS_PSTAT_FORM_WAIT
2 = NWPS_PSTAT_PRINTING
3 = NWPS_PSTAT_PAUSED
4 = NWPS_PSTAT_STOPPED
5 = NWPS_PSTAT_MARK_EJECT
6 = NWPS_PSTAT_READY_TO_DOWN
7 = NWPS_PSTAT_NOT_CONNECTED
8 = NWPS_PSTAT_PRIVATE
```

troubleCode values follow:

```
0 = NWPS_PRINTER_RUNNING
```

```
1 = NWPS_PRINTER_OFFLINE
2 = NWPS_PRINTER_PAPER_OUT
```

serviceMode values follow:

```
0 = NWPS_QUEUE_ONLY
1 = NWPS_QUEUE_BEFORE_FORM
2 = NWPS_FORM_ONLY
3 = NWPS FORM BEFORE QUEUE
```

NWPSComGetPrinterStatus is for operators and users.

NCP Calls

None

See Also

NWPSComAttachToPrintServer, NWPSComLoginToPrintServer

NWPSComGetPrintJobStatus

Retrieves the status of the current print job on the specified printer
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComGetPrintJobStatus (
   NWPS_SPXID_HANDLE   spxID,
   nuint               printerID,
   pnstr8              nServerName,
   pnstr8              queueName,
   pnuint32            jobID,
   pnstr8              jobDescription,
   pnuint              copies,
   pnuint32            printJobSize,
   pnuint              copiesDone,
   pnuint32            bytesDone,
   pnuint              formNumber,
   pnuint               textFlag);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

printerID

(IN) Specifies the printer (0 to maximum number of printers minus 1).

nServerName

(OUT) Points to the name of the NetWare server.

queueName

(OUT) Points to the name of the queue the job came from.

jobID

(OUT) Points to the queue job number.

jobDescription

(OUT) Points to the job description from the queue job header.

copies

(OUT) Points to the number of copies to be printed.

printJobSize

(OUT) Points to the size of each copy requested.

copiesDone

(OUT) Points to the number of copies printed or buffered.

bytesDone

(OUT) Points to the number of bytes already printed in the currently printing copy.

formNumber

(OUT) Points to the number of the current form.

textFlag

(OUT) Points to a value indicating whether to expand tabs: TRUE=(text) expand tabs; FALSE=(byte stream) do not expand tabs.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0 000	NWPSE_SUCCESSFUL
0x0 0ED	NWPSE_CONNECTION_TERMINATED
0x0 103	NWPSE_BINDERY_LOCKED
0x0 301	NWPSE_NOT_ENOUGH_MEMORY
0x0 302	NWPSE_NO_SUCH_PRINTER
0x0 309	NWPSE_NO_JOB_ACTIVE
0x0 30C	NWPSE_DOWN
0x0 30E	NWPSE_NO_RIGHTS

Remarks

printJobSize is the size of the print job in bytes, including a form feed, unless suppress form feed has been set.

copiesDone is the number of copies already printed, including a form feed, unless suppress form feed has been set.

NWPSComGetPrintJobStatus is for operators and users.

NCP Calls

None

See Also

NWPSComAttachToPrintServer, **NWPSComGetPrinterStatus**, **NWPSComLoginToPrintServer**

NWPSComGetPrintServerInfo

Retrieves the current status of the print server and returns information about the services the print server offers

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComGetPrintServerInfo (
   NWPS_SPXID_HANDLE    spxID,
   NWPS_PSInfo NWPTR    psInfo,
   nuint                infoSize);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

psInfo

(OUT) Points to NWPS_PSInfo.

infoSize

(IN) Specifies the size of information requested in NWPS_PSInfo.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0 000	NWPSE_SUCCESSFUL
0x0 0ED	NWPSE_CONNECTION_TERMINATED
0x0 103	NWPSE_BINDERY_LOCKED
0x0 301	NWPSE_NOT_ENOUGH_MEMORY
0x0 30C	NWPSE_DOWN

Remarks

NWPSComGetPrintServerInfo has no user restrictions.

NOTE: If the print server is version 1.1 or newer*futureUse* is 7 bytes long; if the print server is 1.0 or older,*futureUse* is 12 bytes long and *serialNumber* and *serverType* are not used.

NCP Calls

None

See Also

NWPSComAttachToPrintServer, NWPSComLoginToPrintServer

NWPSComGetQueuesServiced

Returns information about one or more queues the specified printer is servicing

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComGetQueuesServiced (
   NWPS_SPXID_HANDLE   spxID,
   nuint               printerID,
   pnuint              sequence,
   pnstr8              nServerName,
   pnstr8              queueName,
   pnuint               priority);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

printerID

(IN) Specifies the printer (0 to maximum number of printers minus 1).

sequence

(IN/OUT) Points to the index. *sequence* must be set to 0 before the first call; it is incremented by **NWPSComGetQueuesServiced** on every return.

nServerName

(OUT) Points to the name of the object's NetWare server.

queueName

(OUT) Points to the name of the queue being serviced.

priority

(OUT) Points to the priority of this queue.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0 000	NWPSE_SUCCESSFUL
0x0 0ED	NWPSE_CONNECTION_TERMINATED
0x0 103	NWPSE_BINDERY_LOCKED
0x0 301	NWPSE_NOT_ENOUGH_MEMORY
0x0 302	NWPSE_NO_SUCH_PRINTER
0x0 307	NWPSE_NOT_CONNECTED
0x0 30C	NWPSE_DOWN
0x0 30D	NWPSE_NOT_IN_LIST
0x0 30E	NWPSE_NO_RIGHTS

Remarks

NWPSComGetQueuesServiced returns NWPSE_NOT_IN_LIST when no queues are left. *priority* is a value between 1 and 10.

NWPSComGetQueuesServiced is for operators and users.

NCP Calls

None

See Also

NWPSComAttachToPrintServer, **NWPSComLoginToPrintServer**, **NWPSComDeleteQueueFromPrinter**, **NWPSComAddQueueToPrinter**

NWPSComLoginToPrintServer

Logs the client in to the print server
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComLoginToPrintServer (
   nuint               connType,
   nuint32             connID,
   NWPS_SPXID_HANDLE   spxID,
   nuint NWFAR        *accessLevel);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

spxID

(IN) Specifies the SPX™ connection to the print server.

accessLevel

(OUT) Points to the client privilege levels from the print server.

Return Values

These are common return values; see 'Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL
0x00ED	NWPSE_CONNECTION_TERMINATED
0x0103	NWPSE_BINDERY_LOCKED

0x0 30C	NWPSE_DOWN
0x0 400	NWPSE_UNABLE_TO_VERIFY_IDENTITY

Remarks

NWPSComLoginToPrintServer gets the client's access rights from the NetWare server Bindery or from the Directory to determine if the client has user or operator rights to the print server. TheSPX™ connection must already be established by calling **NWPSComAttachToPrintServer**.

If *connType* is NWPS_BINDERY_SERVICE, the client must be attached to one of the NetWare servers to which the print server is attached. If *connType* is NWPS_DIRECTORY_SERVICE, the client must be authenticated on the Directory tree.

accessLevel has the following values:

```
0 = NWPS_LIMITED
1 = NWPS_USER
2 = NWPS OPERATOR
```

NWPSComLoginToPrintServer is for anyone.

NCP Calls

None

See Also

NWPSComAttachToPrintServer, **NWPSComLoginToPrintServer**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSComMarkTopOfForm

Sends a line of characters to the specified printer indicating where the top of the page is located

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComMarkTopOfForm (
   NWPS_SPXID_HANDLE   spxID,
   nuint               printerID,
   char                mark);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

printerID

(IN) Specifies the printer (0 to maximum number of printers minus 1).

mark

(IN) Specifies the character to print across the top of the form.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL
0x00ED	NWPSE_CONNECTION_TERMINATED
0x0103	NWPSE_BINDERY_LOCKED
0x0301	NWPSE_NOT_ENOUGH_MEMORY
0x0302	NWPSE_NO_SUCH_PRINTER

0x0 304	NWPSE_PRINTER_BUSY
0x0 307	NWPSE_NOT_CONNECTED
0x0 306	NWPSE_GOING_DOWN
0x0 30C	NWPSE_DOWN
0x0 30E	NWPSE_NO_RIGHTS

Remarks

If the operator chooses an illegal character, an asterisk is substituted. The printer must be inactive for this packet to be successful. If the printer is active, an error is returned.

NWPSComMarkTopOfForm is for operators only.

NCP Calls

None

See Also

NWPSComGetPrinterStatus, **NWPSComAttachToPrintServer**, **NWPSComLoginToPrintServer**

NWPSComPrintServerRequest

Transmits a request packet to the print server using theSPX™ connection, and receives the corresponding reply before returning it

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComPrintServerRequest (
   NWPS_SPXID_HANDLE   spxID,
   nptr                reqBuffer,
   int                 reqSize,
   nptr                repBuffer,
   int                 repSize);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

reqBuffer

(IN) Points to the address of the request packet buffer.

reqSize

(IN) Specifies the length of the request packet.

repBuffer

(OUT) Points to the address of the reply packet buffer.

repSize

(OUT) Receives the length of the expected reply.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL
0x00ED	NWPSE_CONNECTION_TERMINATED

Remarks

NWPSComPrintServerRequest allows for expansion of the supported services without adding more calls to nwps_com.h.

reqBuffer contains the request, formated as indicated in nwps_pkt.h. When the last data item in the packet is a zero-terminated ASCII string, only the length of the packet up to and including the zero needs to be specified for *reqSize*.

repBuffer need only be as large as *repSize*. The data returned from the print server is the full size of the packet described in nwps_pkt.h, but only the amount of data specified by *repSize* is copied into the caller's *repBuffer*.

An NWPSE_SUCCESSFUL return code indicates a reply was received for the request. If this is the case, the user must check the reply packet's *returnCode*. This field may contain errors from the print server ranging from 0x100 to 0x600, depending on the nature of the problem. See Appendix A, Error Control Codes, for a complete list of return codes.

NCP Calls

None

NWPSComRequestRemotePrinter

Requests the print server to reserve the specified printer
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComRequestRemotePrinter (
   NWPS_SPXID_HANDLE   spxID,
   nuint               printerID,
   NWPS_NInfo NWPTR    info);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

printerID

(IN) Specifies the printer (0 to maximum number of printers minus 1).

info

(IN) Points to NWPS_RInfo.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL
0x00ED	NWPSE_CONNECTION_TERMINATED
0x0301	NWPSE_NOT_ENOUGH_MEMORY
0x0302	NWPSE_NO_SUCH_PRINTER
0x0308	NWPSE_ALREADY_IN_USE
0x0	NWPSE_DOWN

30C	

Remarks

NWPSComRequestRemotePrinter applies only to printers that are remote in relation to the print server (user-loadable).

The client should first call **NWPSComGetNextRemotePrinter** to find out which printer numbers are available.

NCP Calls

None

See Also

NWPSComAttachToPrintServer, **NWPSComGetNextRemotePrinter**, **NWPSComLoginToPrintServer**

NWPSComRewindPrintJob

Rewinds the current print job the specified number of pages or bytes and restarts it from that point

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComRewindPrintJob (
   NWPS_SPXID_HANDLE   spxID,
   nuint               printerID,
   nuint               byPage,
   nuint               relative,
   nuint               copy,
   nuint32             offset);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

printerID

(IN) Specifies the printer (0 to maximum number of printers minus 1).

byPage

(IN) Specifies whether to rewind by page: FALSE = by byte; TRUE = by page.

relative

(IN) Specifies whether to rewind an absolute value: FALSE = absolute; TRUE = relative.

copy

(IN) Specifies the copy number to rewind to. If *relative* = TRUE, *copy* is not used.

offset

(IN) Specifies the page number or byte offset to restart from.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK*

Getting Started for more information.

0x0 000	NWPSE_SUCCESSFUL
0x0 0ED	NWPSE_CONNECTION_TERMINATED
0x0 301	NWPSE_NOT_ENOUGH_MEMORY
0x0 302	NWPSE_NO_SUCH_PRINTER
0x0 304	NWPSE_PRINTER_BUSY
0x0 305	NWPSE_NOT_CONNECTED
0x0 309	NWPSE_NO_JOB_ACTIVE
0x0 30C	NWPSE_DOWN
0x0 30E	NWPSE_NO_RIGHTS

Remarks

If absolute is selected, the printer rewinds to the specified page number or byte offset. If relative is selected, the printer rewinds the specified number of pages or bytes from the current page or byte.

If *byPage* is TRUE and *relative* is FALSE, *offset* indicates the page number to restart at. If *relative* is TRUE, *offset* indicates the number of pages to rewind forward or backward.

If *byPage* is FALSE and *relative* is FALSE, *offset* indicates the byte offset to restart at. If *relative* is TRUE, *offset* indicates the number of bytes to rewind forward or backward.

NWPSComRewindPrintJob is for operators only.

NCP Calls

None

See Also

NWPSComAttachToPrintServer, **NWPSComGetPrintJobStatus**, **NWPSComLoginToPrintServer**

NWPSComSetMountedForm

Informs the print server the specified form has been mounted on the indicated printer

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComSetMountedForm (
   NWPS_SPXID_HANDLE   spxID,
   nuint               printerID,
   nuint               formNumber);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

printerID

(IN) Specifies the printer (0 to maximum number of printers minus 1).

formNumber

(IN) Specifies the form number.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL
0x00ED	NWPSE_CONNECTION_TERMINATED
0x0301	NWPSE_NOT_ENOUGH_MEMORY
0x0302	NWPSE_NO_SUCH_PRINTER
0x030C	NWPSE_DOWN

0x0 30E	NWPSE_NO_RIGHTS

Remarks

The printer number must refer to a valid printer.

NWPSComSetMountedForm is for operators only.

NCP Calls

None

See Also

NWPSComAttachToPrintServer, **NWPSComGetPrintServerInfo**, **NWPSComLoginToPrintServer**

NWPSComSetRemoteMode

Sets the printer mode
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComSetRemoteMode (
   NWPS_SPXID_HANDLE   spxID,
   nuint               printerID,
   nuint               newMode);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

printerID

(IN) Specifies the printer (0 to maximum number of printers minus 1).

newMode

(IN) Specifies the mode to set the printer to: 0 = NWPS_PRINTER_SHARED; 1 = NWPS_PRINTER_PRIVATE.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL
0x00ED	NWPSE_CONNECTION_TERMINATED
0x0103	NWPSE_BINDERY_LOCKED
0x0301	NWPSE_NOT_ENOUGH_MEMORY
0x0302	NWPSE_NO_SUCH_PRINTER

0x0 303	NWPSE_INVALID_PARAMETER
0x0 304	NWPSE_PRINTER_BUSY
0x0 307	NWPSE_NOT_CONNECTED
0x0 30C	NWPSE_DOWN
0x0 30E	NWPSE_NO_RIGHTS
0x0 401	NWPSE_NOT_USER_START_PRINTER

Remarks

NWPSComSetRemoteMode applies only to printers that are remote in relation to the print server (user-loadable).

The packet is sent on the client socket. When *newMode* is NWPS_PRINTER_SHARED, the printer is shared with the network. When *newMode* is NWPS_PRINTER_PRIVATE, the printer is private to the workstation.

NCP Calls

None

See Also

NWPSComAttachToPrintServer, **NWPSComGetNextRemotePrinter**, **NWPSComLoginToPrintServer**, **NWPSComRequestRemotePrinter**

NWPSComStartPrinter

Restarts a stopped printer
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComStartPrinter (
   NWPS_SPXID_HANDLE   spxID,
   nuint               printerID);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

printerID

(IN) Specifies the printer (0 to maximum number of printers minus 1).

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0 000	NWPSE_SUCCESSFUL
0x0 0ED	NWPSE_CONNECTION_TERMINATED
0x0 301	NWPSE_NOT_ENOUGH_MEMORY
0x0 302	NWPSE_NO_SUCH_PRINTER
0x0 307	NWPSE_NOT_CONNECTED
0x0 30C	NWPSE_DOWN
0x0 30E	NWPSE_NO_RIGHTS

Remarks

If the print job is paused when the printer is stopped, the print job resumes when **NWPSComStartPrinter** is issued. If the print job was not paused, the print server requests the next job. If the printer is not stopped when **NWPSComStartPrinter** is issued, it returns NWPSE_SUCCESSFUL.

NWPSComStartPrinter is for operators only.

NCP Calls

None

See Also

NWPSComAttachToPrintServer, **NWPSComGetPrinterStatus**, **NWPSComLoginToPrintServer**, **NWPSComStopPrinter**

NWPSComStopPrinter

Stops the specified printer
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_com.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSComStopPrinter (
   NWPS_SPXID_HANDLE   spxID,
   nuint               printerID,
   nuint               jobOutcome);
```

Parameters

spxID

(IN) Specifies the SPX™ connection to the print server.

printerID

(IN) Specifies the printer (0 to maximum number of printers minus 1).

jobOutcome

(IN) Specifies what to do after stopping the printer: 1 = NWPS_RETURN_TO_QUEUE; 2 = NWPS_THROW_AWAY.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL
0x00ED	NWPSE_CONNECTION_TERMINATED
0x0301	NWPSE_NOT_ENOUGH_MEMORY
0x0302	NWPSE_NO_SUCH_PRINTER
0x0307	NWPSE_NOT_CONNECTED

0x0 30C	NWPSE_NO_RIGHTS
0x0 30E	NWPSE_DOWN

Remarks

If the job returns to the queue, it restarts from the beginning. If the printer is already stopped and this packet is issued, it returns NWPSE_SUCCESSFUL.

NWPSComStopPrinter is for operators only.

NCP Calls

None

See Also

NWPSComAttachToPrintServer, **NWPSComGetPrinterStatus**, **NWPSComLoginToPrintServer**, **NWPSComStartPrinter**

NWPSDeRegisterLibraryClient

Closes NWPS functions in NLM applications
Local Servers: N/A
Remote Servers: N/A
NetWare Server: 3.x, 4.x
Platform: NLM
Service: Print Server

Syntax

```
#include <nwps_nlm.h>
#include <nwps_err.h>

int NWPSDeRegisterLibraryClient (
   void);
```

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x7 7B0	NWPSE_UNREGISTERED_THREAD
0x7 7B1	NWPSE_UNREGISTERED_NLM

Remarks

NWPSDeRegisterLibraryClient must be called before an NLM calling NWPS functions is unloaded.

NWPSRV3X.NLM must be used for **NWPSDeRegisterLibraryClient** to be called in a 3.x environment.

NCP Calls

None

NWPSGetLibraryVersion

Provides the version number of NWPSRV.NLM currently loaded on the server

Local Servers: N/A

Remote Servers: N/A

NetWare Server: 3.x, 4.x

Platform: NLM

Service: Print Server

Syntax

```
#include <nwps_nlm.h>

N_EXTERN_LIBRARY(void) NWPSGetLibraryVersion (
   pnuint8   majorVersion;
   pnuint8   minorVersion;
   pnuint8   revisionLevel;
   pnuint8   betaReleaseLevel);
```

Parameters

majorVersion

(OUT) Points to the major version of NWPSRV.NLM currently loaded on the server.

minorVersion

(OUT) Points to the minor version of NWPSRV.NLM currently loaded on the server.

revisionLevel

(OUT) Points to the revision level of NWPSRV.NLM currently loaded on the server.

betaReleaseLevel

(OUT) Points to the beta release level of NWPSRV.NLM currently loaded on the server.

Return Values

None

Remarks

NWPSGetLIbraryVersion provides the same information to an NLM that is available on screen through the NDIR command with a /ver switch. However, with **NWPSGetLibraryVersion**, you can get individual

components of the version designation as desired.

To obtain the desired information, pass the address of an allocated BYTE sized memory location to the appropriate parameters. **NWPSGetLIbraryVersion** will return the information designated by that parameter to the address(es) you pass. If you want no information for one or more parameters, simply pass NULL to each such parameter, and **NWPSGetLIbraryVersion** will not return anything to them.

NCP Calls

None

NWPSJobDelete

Removes a print job record from PRINTCON
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_job.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSJobDelete (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *dbOwner,
   char NWFAR   *pJobName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

dbOwner

(IN) Points to the user name of the owner of the database where the print job is defined.

pJobName

(IN) Points to the print job record name to delete.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL
0x7770	NWPSE_BAD_VERSION
0x7	NWPSE_ERROR_OPENING_DB

774	
0x7775	NWPSE_ERROR_READING_DB
0x7777	NWPSE_ERROR_WRITING_DB
0x7779	NWPSE_INTERNAL_ERROR
0x777A	NWPSE_JOB_NOT_FOUND

Remarks

Since multiple PRINTCON databases exist, *dbOwner* specifies which database is affected. If the print job does not exist in the database, NWPSE_SUCCESSFUL is returned.

If *dbOwner* is NWPS_DBOWNER_PUBLIC, the public database of the NetWare™ server or those of the nearest organizational unit on the directory is affected. This is not valid for NWPS_BINDERY_PRE_40.

In NWPS_BINDERY_SERVICE and NWPS_BINDERY_SERVICE_PRE_40 mode, *dbOwner* is a user on the NetWare server.

In NWPS_DIRECTORY_SERVICE mode, *dbOwner* is a user or an organizational unit in the Directory.

NCP Calls

None

See Also

NWPSJobWrite, **NWPSJobRead**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSJobEndNextJob

Frees up memory and does other cleanup associated with the scan started by **NWPSJobGetFirstJob**

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_cfg.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSJobEndNextJob (
   NWPSListHandle   handle);
```

Parameters

handle

(IN) Specifies the value returned from **NWPSJobGetFirstJob**.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

NWPSJobEndNextJob is required only if the handle returned from **NWPSJobGetFirstJob** was non-zero. Calling **NWPSJobEndNextJob** with a handle of zero has no effect.

NCP Calls

None

See Also

NWPSJobGetFirstJob, **NWPSJobGetNextJob**

NWPSJobGetDefault

Returns the contents of the default print job record of the directory or NetWare server specified by *connID*

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_job.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSJobGetDefault (
   WORD                  connType,
   DWORD                 connID,
   WORD                  searchFlag,
   char NWFAR           *dbOwner,
   char NWFAR           *pJobName,
   NWPS_Job_Rec NWFAR  *pJobRecord);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

searchFlag

(IN) Specifies the database to be searched.

dbOwner

(IN/OUT) Points to the username of the owner of the database to search (optional) upon input. Points to the owner of the database where *pJobName* is actually located upon output. Its maximum length is MAX_DN_BYTES.

pJobName

(OUT) Points to the name of the default print job found. Its maximum length is NWPS_JOB_NAME_SIZE.

pJobRecord

(OUT) Points to where the content of the default print job record is stored.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL
0x7770	NWPSE_BAD_VERSION
0x7773	NWPSE_ERROR_GETTING_DEFAULT
0x7774	NWPSE_ERROR_OPENING_DB
0x7775	NWPSE_ERROR_READING_DB
0x7776	NWPSE_ERROR_READING_RECORD
0x7779	NWPSE_INTERNAL_ERROR
0x777B	NWPSE_DEFAULT_SPECIFIED

Remarks

searchFlag is defined as follows:

```
0 = NWPS_EXTENDED_SEARCH
1 = NWPS_SINGLE_SEARCH
2 = NWPS_LIMITED_SEARCH
```

searchFlag indicates if only the specified database is to be searched (NWPS_SINGLE_SEARCH), if all databases starting at the specified one are to be searched (NWPS_EXTENDED_SEARCH), or if only the first database found is to be searched (NWPS_LIMITED_SEARCH).

You can set *dbOwner* to the username of the owner of the database to be searched, or to NULL if all databases are to be searched starting at the current user's database. *dbOwner* returns the actual location where the default print job was found.

When a default print job is found, its name is returned in *pJobName*, and the contents of the print job record are copied into the buffer pointed to by *pJobRecord*.

If no default print job is found, **NWPSJobGetDefault** returns an empty string in *pJobName* and a non-zero completion code.

To get the name of the default print job only, without the contents of the print job record, pass NULL in *pJobRecord*.

NCP Calls

None

See Also

NWPSJobSetDefault, **NWPSJobWrite**, **NWPSJobRead**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSJobGetFirstJob

Retrieves the first job name and owner for a defined print job configuration in the Bindery or Directory context

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_job.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSJobGetFirstJob (
   WORD                    connType,
   DWORD                   connID,
   WORD                    searchFlag,
   NWPSListHandle NWFAR   *handle,
   char NWFAR             *dbOwner,
   char NWFAR             *pJobName,
   WORD NWFAR            *defaultPJ);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection identifier.

searchFlag

(IN) Specifies the search flag.

handle

(OUT) Points to the value to be passed to **NWPSJobGetNextJob** and **NWPSJobEndNextJob**.

dbOwner

(IN/OUT) Points to the start point of the search for a print job (optional) upon input. Points to the actual location where the print job was found upon output. Its maximum length is MAX_DN_BYTES.

pJobName

(OUT) Points to the name of the first print job record found. Its maximum length is NWPS_JOB_NAME_SIZE.

defaultPJ

(OUT) Points to the single_byte default print job flag defined as follows:

```
NWPS_DOMINANT_DEFAULT_JOB     1
NWPS_CURRENT_DB_DEFAULT_JOB   2
                              0
or any combination may be returned.
```

Return Values

0x0000	Successful
0x7771	End of Job Names
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

NWPSJobGetFirstJob allocates memory and opens files that must be respectively deallocated and closed by calling **NWPSJobEndNextJob**. If the completion code is non-zero, the value returned for the handle is zero. In this case, calling **NWPSJobEndNextJob** is not required.

jobSearch is defined as follows:

```
0 = NWPS_EXTENDED_SEARCH
1 = NWPS_SINGLE_SEARCH
2 = NWPS_LIMITED_SEARCH
```

searchFlag indicates if only the specified database is to be searched (NWPS_SINGLE_SEARCH), or if all relative databases starting at the specified one are to be searched (NWPS_EXTENDED_SEARCH). When NWPS_LIMITED_SEARCH is specified, **NWPSJobGetFirstJob** looks only at the first location, plus an optional second level if the first level is not an Organization or Organizational Unit.

dbOwner returns the directory object name or Bindery user name of the owner of the database to be searched. You can set it to NULL if all relative databases are to be searched starting at the current user's database.

Each function returns the name of the next print job record found and places it in *pJobName*.

NCP Calls

None

Print Service Group

See Also

NWPSJobEndNextJob, **NWPSJobGetNextJob**, **NWPSJobRead**

NWPSJobGetNextJob

Retrieves the next job name and owner for a defined print job configuration in the Bindery or Directory context

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_job.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSJobGetNextJob (
   NWPSListHandle   handle,
   char NWFAR      *dbOwner,
   char NWFAR      *pJobName,
   WORD NWFAR      *defaultPJ);
```

Parameters

handle

(IN) Specifies the value returned from **NWPSJobGetFirstJob**.

dbOwner

(IN/OUT) Points to the username of the owner of the database (optional). Its maximum length is MAX_DN_BYTES.

pJobName

(OUT) Points to the print job name found. Its maximum length is NWPS_JOB_NAME_SIZE.

defaultPJ

(OUT) Points to single-byte default job flag, defined as follows:

```
NWPS_DOMINANT_DEFAULT_JOB     1
NWPS_CURRENT_DB_DEFAULT_JOB   2
or any combination may be returned
```

Return Values

0x0000	Successful
0x7771	End of Job Names
-1	General Error
other	Bindery or Directory Services Errors

values	

Remarks

dbOwner returns the directory object name or Bindery user name of the owner of the database to be searched. You can set it to NULL if all relative databases are to be searched starting at the current user's database.

If the handle is zero, NWPSE_END_SCAN is returned, indicating the end of the list of job names.

NCP Calls

None

See Also

NWPSJobGetFirstJob, **NWPSJobEndNextJob**, **NWPSJobRead**

NWPSJobInit

Initializes a print job record with default values
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_job.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSJobInit (
   NWPS_Job_Rec NWFAR  *pJobRecord);
```

Parameters

pjobRecord

(IN) Points to NWPS_Job_Rec to complete with defaults.

Return Values

0	Success
-1	Failure

NCP Calls

None

See Also

NWPSJobSetDefault, **NWPSJobWrite**, **NWPSJobSet**

NWPSJobRead

Searches for a print job record in the PRINTCON database
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_job.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSJobRead (
   WORD                  connType,
   DWORD                 connID,
   char NWFAR           *dbOwner,
   char NWFAR           *pJobName,
   NWPS_Job_Rec NWFAR   *pJobRecord);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

dbOwner

(IN/OUT) Points to the user name of the owner of the database (optional).

pJobName

(IN/OUT) Points to the name of the print job to read.

pJobRecord

(OUT) Points to NWPS_Job_Rec where the print job record is stored.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL

0x7770	NWPSE_BAD_VERSION
0x7773	NWPSE_ERROR_GETTING_DEFAULT
0x7774	NWPSE_ERROR_OPENING_DB
0x7775	NWPSE_ERROR_READING_DB
0x7776	NWPSE_ERROR_READING_RECORD
0x7779	NWPSE_INTERNAL_ERROR
0x777B	NWPSE_NO_DEFAULT_SPECIFIED

Remarks

dbOwner is the user name of the owner of the database to be searched, or it is NULL if all databases are to be searched starting at the current user's database. If a NULL is used, *dbOwner* returns the name of the job owner.

If *dbOwner* is NWPS_DBOWNER_PUBLIC, the public database of the NetWare server or the database of the nearest organizational unit on the directory is affected.

In NWPS_BINDERY_SERVICE or NWPS_BINDERY_SERVICE_PRE_40 mode, *dbOwner* is a user on the NetWare server. (NWPS_DBOWNER_PUBLIC is not valid for NWPS_BINDERY_SERVICE_PRE_40.)

In NWPS_DIRECTORY_SERVICE mode, *dbOwner* is a user or an organizational unit in the directory.

NCP Calls

None

See Also

NWPSJobWrite, **NWPSJobGetDefault**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSJobScan (obsolete 6/96)

Iteratively scans the public and private PRINTCONs of the specified Directory or NetWare server to get a list of currently defined print job configurations but is now obsolete. Call **NWPSJobGetFirstJob**, **NWPSJobGetNextJob**, or **NWPSJobEndNextJob** instead.

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT

Service: Print Server

Syntax

```
#include <nwps_job.h>
or
#include <nwpsrv.h>

NWCCODE NWAPI NWPSJobScan
   (WORD          connType,
    DWORD         connID,
    WORD NWFAR   *pJSequence,
    WORD          searchFlag,
    char NWFAR   *dbOwner,
    char NWFAR   *pJobName,
    WORD NWFAR   *defaultPJ);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

pJSequence

(IN/OUT) Points to the print job sequence; it increments with each **NWPSJobScan (obsolete 6/96)** call, beginning with -1.

searchFlag

(IN) Specifies the database to search.

dbOwner

(IN/OUT) Points to the start point of the search for a print job (optional) upon input. Points to the actual location where the print job was found upon output. Its maximum length is NWPS_JOB_NAME_SIZE.

pJobName

(IN/OUT) Points to the name of each print job record scanned. Its maximum length is NWPS_JOB_NAME_SIZE.

defaultPJ

(OUT) Points to the default job:

```
NWPS_DOMINANT_DEFAULT_JOB     1
NWPS_CURRENT_DB_DEFAULT_JOB   2
or any combination may be returned
```

Return Values

0x0000	NWPSE_SUCCESSFUL
0x7770	NWPSE_BAD_VERSION
0x7771	NWPSE_END_SCAN
0x7773	NWPSE_ERROR_GETTING_DEFAULT
0x7774	NWPSE_ERROR_OPENING_DB
0x7775	NWPSE_ERROR_READING_DB
0x7779	NWPSE_INTERNAL_ERROR

Remarks

pJSequence must be set to -1 the first time**NWPSJobScan (obsolete 6/96)** is called to begin a new sequence.

searchFlag can have the following values:

```
0 = NWPS_EXTENDED_SEARCH
1 = NWPS_SINGLE_SEARCH
2 = NWPS_LIMITED_SEARCH
```

searchFlag indicates if only the specified database is to be searched (NWPS_SINGLE_SEARCH), or if all relative databases starting at the specified one are to be searched (NWPS_EXTENDED_SEARCH). When NWPS_LIMITED_SEARCH is specified, **NWPSJobScan (obsolete 6/96)** looks only at the first location, plus an optional second level if the first level is not an Organization or Organizational Unit.

dbOwner returns the directory object name or bindery username of the owner of the database to be searched. You can set it to NULL if all

relative databases are to be searched starting at the current user's database.

Each function returns the name of the next print job record found and places it in *pJobName*.

A one or a zero is returned in *defaultPJ*, indicating whether the print job is the default (1) or not the default (0).

NCP Calls

None

See Also

NWPSJobGetDefault, **NWPSJobRead**, **NWPSJobGetFirstJob**, **NWPSJobGetNextJob**, **NWPSJobEndNextJob**, **NWCCScanConnRefs**, **NWCCOpenConnByRef**, **NWDSCreateContextHandle**

NWPSJobSet

Sets a print job record with defined values
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_job.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSJobSet (
   WORD                  connType,
   NWPS_Job_Rec NWFAR   *pJobRecord,
   char NWFAR           *formName,
   char NWFAR           *deviceName,
   char NWFAR           *modeName,
   char NWFAR           *bannerName,
   char NWFAR           *jobName,
   char NWFAR           *bindNServr,
   char NWFAR           *bindQueue,
   char NWFAR           *bindPserver,
   WORD                  dsUseQueueName,
   char NWFAR          *dsObjectName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICES_PRE_40, or NWPS_DIRECTORY_SERVICES.

pJobRecord

Points to NWPS_Job_Rec.

formName

Points to the form name.

modeName

Points to the mode name.

bannerName

Points to the banner name.

jobName

Points to the job name.

bindNServer

Points to the Bindery NetWare server name.

bindQueue

Points to the Bindery queue name.

bindPServer

Points to the Bindery print server name.

dsUseQueueName

Specifies which queue to use: TRUE=next field (*dsObjectName*) is the queue.

dsObjectName

Points to the queue or printer name.

Return Values

0	Success
-1	Failure

Remarks

pJobRecord should have been initialized by calling **NWPSJobInit**.

NCP Calls

None

See Also

NWPSJobInit, **NWPSJobSetDefault**, **NWPSJobWrite**

NWPSJobSetDefault

Sets or resets the specified default print job record
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_job.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSJobSetDefault (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *dbOwner,
   char NWFAR   *pJobName,
   char NWFAR   *pJobOwner);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

dbOwner

(IN) Points to the username of the owner of the database.

pJobName

(IN) Points to the name of the print job to set or unset as the default.

pJobOwner

(IN) Points to the owner of the database where *pJobName* is defined.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL
0x7	NWPSE_BAD_VERSION

770	
0x7774	NWPSE_ERROR_OPENING_DB
0x7775	NWPSE_ERROR_READING_DB
0x7777	NWPSE_ERROR_WRITING_DB
0x7779	NWPSE_INTERNAL_ERROR
0x777A	NWPSE_JOB_NOT_FOUND

Remarks

dbOwner is the name of the user whose database is to be modified.

pJobName is the new default print job for the *dbOwner*'s database or NULL to remove the current default print job.

pJobOwner points to the database where *pJobName* is defined. No attempt is made to verify that the print job exists in the *pJobOwner* database.

NCP Calls

None

See Also

NWPSJobGetDefault, **NWPSJobWrite**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSJobWrite

Creates or modifies the print job record specified by *pJobName* in the PRINTCON database

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_job.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSJobWrite (
   WORD                  connType,
   DWORD                 connID,
   char NWFAR           *dbOwner,
   char NWFAR           *pJobName,
   NWPS_Job_Rec NWFAR   *pJobRecord);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

dbOwner

(IN) Points to the username of the owner of the database.

pJobName

(IN) Points to the name of the print job record to be written.

pJobRecord

(IN) Points to NWPS_Job_Rec.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL

0x7770	NWPSE_ERROR_BAD_VERSION
0x7772	NWPSE_ERROR_EXPANDING_DB
0x7774	NWPSE_ERROR_OPENING_DB
0x7777	NWPSE_ERROR_WRITING_DB
0x7778	NWPSE_ERROR_WRITING_RECORD
0x7779	NWPSE_INTERNAL_ERROR
0x777A	NWPSE_JOB_NOT_FOUND

Remarks

dbOwner is the username of the owner whose database is written to.

If a print job record with the same name already exists, it is overwritten with the data pointed to by *pJobRecord*. Otherwise, a new record is created.

NCP Calls

None

See Also

NWPSJobRead, **NWPSJobSetDefault**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSPdfAddDevice

Adds a device to the PRINTDEF database
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfAddDevice (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *deviceName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

deviceName

(IN) Points to the new device name.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

The device is created without any functions or modes. Once a device is defined, functions and modes can be added to the device.

NCP Calls

None

See Also

NWPSPdfDeleteDevice, **NWPSPdfReadDevice**, **NWPSPdfUpdateDevice**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSPdfAddForm

Adds a form definition to PRINTDEF database
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfAddForm (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *formName,
   WORD          formNumber,
   WORD          formLength,
   WORD          formWidth);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

formName

(IN) Points to the form name.

formNumber

(IN) Specifies the unique form identifier used by applications.

formLength

(IN) Specifies the number of lines on a text page.

formWidth

(IN) Specifies the number of characters per line on a text page.

Return Values

0x0000	Successful
-1	General Error

other values	Bindery or Directory Services Errors

Remarks

Forms are independent of printers and are unique on each file server or context.

NCP Calls

None

See Also

NWPSPdfDeleteForm, **NWPSPdfReadForm**, **NWPSPdfUpdateForm**

NWPSPdfAddFunction

Adds a function string to PRINTDEF database
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfAddFunction (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *deviceName,
   char NWFAR   *funcName,
   WORD          funcSize,
   BYTE NWFAR   *funcString);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

deviceName

(IN) Points to the associated device name.

funcName

(IN) Points to the function name to add.

funcSize

(IN) Specifies the count of bytes in the function to add.

funcString

(IN) Points to the list of bytes for the new function.

Return Values

0x0000	Successful
-1	General Error

other values	Bindery or Directory Services Errors

Remarks

funcSizes should be set to the number of bytes in *funcString*.

NCP Calls

None

See Also

NWPSPdfAddDevice, **NWPSPdfDeleteFunction**, **NWPSPdfReadFunction**, **NWPSPdfUpdateFunction**, **NWPSPdfAddMode**, **NWPSPdfAddModeFunction**

NWPSPdfAddMode

Adds a new mode to a device in PRINTDEF database
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfAddMode (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *deviceName,
   char NWFAR   *modeName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

deviceName

(IN) Points to the associated device name.

modeName

(IN) Points to the new mode name.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

NCP Calls

None

None

See Also

NWPSPdfAddDevice, **NWPSPdfDeleteMode**, **NWPSPdfReadMode**, **NWPSPdfUpdateMode**, **NWPSPdfAddFunction**, **NWPSPdfAddModeFunction**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSPdfAddModeFunction

Adds a previously defined function to a previously defined mode list in PRINTDEF database

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfAddModeFunction (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *deviceName,
   char NWFAR   *modeName,
   char NWFAR   *funcName,
   WORD          location);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

deviceName

(IN) Points to the associated device name.

modeName

(IN) Points to the mode name.

funcName

(IN) Points to the function name.

location

(IN) Specifies the insertion point or -1.

Return Values

0x0000	Successful

-1	General Error
other values	Bindery or Directory Services Errors

Remarks

The new function is added at the specified location in the mode's list, with 0 being the first location. If *location* is past the end of the list or is -1, the function is added as the last function in the mode's list.

NCP Calls

None

See Also

NWPSPdfAddDevice, **NWPSPdfAddFunction**, **NWPSPdfDeleteModeFunction**, **NWPSPdfReadModeFunction**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSPdfDeleteDevice

Deletes a device from the PRINTDEF database
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfDeleteDevice (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *deviceName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

deviceName

(IN) Points to the device name to delete.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

NWPSPdfDeleteDevice also deletes the functions and modes of the device.

NCP Calls

None

See Also

NWPSPdfAddDevice, **NWPSPdfReadDevice**, **NWPSPdfUpdateDevice**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSPdfDeleteForm

Deletes a form from the PRINTDEF database
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfDeleteForm (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *formName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

formName

(IN) Points to the form name to delete.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

NCP Calls

None

See Also

NWPSPdfAddForm, NWPSPdfReadForm, NWPSPdfUpdateForm, NWCCOpenConnByName, NWDSCreateContextHandle

NWPSPdfDeleteFunction

Deletes a function string from the associated device in the PRINTDEF database

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE)NWPSPdfDeleteFunction (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *deviceName,
   char NWFAR   *funcName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

deviceName

(IN) Points to the associated device name.

funcName

(IN) Points to the name of the function to remove.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

NWPSPdfDeleteFunction also deletes the mode function relationship if the function is referred to in any mode.

NCP Calls

None

See Also

NWPSPdfAddDevice, **NWPSPdfAddFunction**, **NWPSPdfReadFunction**, **NWPSPdfUpdateFunction**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSPdfDeleteMode

Deletes a mode from the associated device in the PRINTDEF database
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfDeleteMode (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *deviceName,
   char NWFAR   *modeName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

deviceName

(IN) Points to the associated device name.

modeName

(IN) Points to the name of the mode to delete.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

When the mode is deleted, the functions are left intact. Functions of modes must be removed by calling **NWPSPdfDeleteFunction**.

NCP Calls

None

See Also

NWPSPdfAddMode, **NWPSPdfReadMode**, **NWPSPdfUpdateMode**, **NWPSPdfDeleteFunction**, **NWPSPdfDeleteModeFunction**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSPdfDeleteModeFunction

Deletes a function from a defined mode list
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfDeleteModeFunction (
   WORD          connType,
   DWORD         connID,
   DWORD         sequence
   char NWFAR   *deviceName,
   char NWFAR   *modeName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

sequence

(IN) Specifies the number of the function to remove from the group.

deviceName

(IN) Points to the associated device name.

modeName

(IN) Points to the mode name.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

NWPSPdfDeleteModeFunction deletes the mode-to-function relationship but leaves the function string and the mode intact.

sequence is the position (zero-relative) of the function to be deleted. This facilitates deletion of the second occurrence of a function within the given mode.

NCP Calls

None

See Also

NWPSPdfAddModeFunction, **NWPSPdfDeleteFunction**, **NWPSPdfDeleteMode**, **NWPSPdfReadModeFunction**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSPdfEndNext

Frees resources allocated by the NWPSPdfGetNext* functions
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

NWCCODE NWAPI NWPSPdfEndNext (
   NWPSListHandle   handle);
```

Parameters

handle

(IN) Specifies the NWPSListHandle initialized by previously calling the following functions:

NWPSPdfGetFirstDevice
NWPSPdfGetFirstForm
NWPSPdfGetFirstFunction
NWPSPdfGetFirstMode
NWPSPdfGetFirstModeFunction

Return Values

0x0000	Successful
0xFFFF	General Error
other values	Bindery or Directory Services Errors

NCP Calls

None

See Also

NWPSPdfGetFirstDevice, **NWPSPdfGetFirstForm**, **NWPSPdfGetFirstFunction**, **NWPSPdfGetFirstMode**,

NWPSPdfGetFirstModeFunction, NWPSPdfGetNextDevice, NWPSPdfGetNextForm, NWPSPdfGetNextFunction, NWPSPdfGetNextMode, NWPSPdfGetNextModeFunction

NWPSPdfExportDevice

Exports a device to a PDF file from the database
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfExportDevice (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *fileName,
   char NWFAR   *deviceName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

fileName

(IN) Points to the name of the .PDF file.

deviceName

(IN) Points to the name of the device to export.

Return Values

These are common return values; see 'Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL
0x7774	NWPSE_ERROR_OPENING_DB
0x7775	NWPSE_ERROR_READING_DB

0x7777	NWPSE_ERROR_WRITING_DB
0x777C	NWPSE_OUT_OF_MEMORY

Remarks

The *fileName* should be in one of the following forms:

```
\\<file server><volume><path><file name>[.PDF] <volume>:<path><file nam
```

If *fileName* is NULL, the file **<**device name**>**[.PDF] is created in the local directory.

Every PDF export file also has a date code associated with it. The date is set when the file is created. See **NWPSPdfSetImportDate** and **NWPSPdfGetImportDate**.

NCP Calls

None

See Also

NWPSPdfGetImportDate, **NWPSPdfImportDevice**, **NWPSPdfSetImportDate**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSPdfGetFirstDevice

Finds a device in the PRINTDEF database
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

NWCCODE NWAPI NWPSPdfGetFirstDevice (
   nuint                 connType,
   nuint32               connID,
   NWPSListHandle NWPTR  handle,
   pnstr8                deviceName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier depending on the *connType* parameter.

handle

(IN/OUT) Points to the address where a NWPSListHandle is stored (used when calling the **NWPSPdfGetNextDevice** and **NWPSPdfEndNext** functions).

deviceName

(OUT) Points to the next device name.

Return Values

0x0000	Successful
0xFFFF	General Error
other values	Bindery or Directory Services Errors

NCP Calls

None

See Also

NWCCScanConnRefs, **NWCCOpenConnByRef**, **NWDSCreateContextHandle**, **NWPSPdfAddDevice**, **NWPSPdfDeleteDevice**, **NWPSPdfEndNext**, **NWPSPdfGetNextDevice**, **NWPSPdfReadDevice**, **NWPSPdfUpdateDevice**

NWPSPdfGetFirstForm

Finds a form in the PRINTDEF database
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfGetFirstForm (
   nuint                  connType,
   nuint32                connId,
   NWPSListHandle NWPTR   handle,
   pnstr8                 formName,
   pnuint                 formNumber,
   pnuint                 formLength,
   pnuint                 formWidth);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connId

(IN) Specifies the connection or context identifier depending on the *connType* parameter.

handle

(IN/OUT) Points to the address where a NWPSListHandle is stored (used when calling the **NWSPDGetNextForm** and **NWPSPdfEndNext** functions.

formName

(OUT) Points to the next form name.

formNumber

(OUT) Points to the address of the form number (optional).

formLength

(OUT) Points to the address of the form length (optional).

formWidth

(OUT) Points to the address for the form width (optional).

Return Values

0x0000	Successful
0xFFFF	General Error
other values	Errors in Bindery or Directory Services

NCP Calls

None

See Also

NWPSPdfEndNext, **NWPSPdfExportDevice**, **NWPSPdfGetNextForm**, **NWPSPdfImportDevice**, **NWPSPdfSetImportDate**

NWPSPdfGetFirstFunction

Finds a function in the PRINTDEF database
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

NWCCODE NWAPI NWPSPdfGetFirstFunction (
   nuint                    connType,
   nuint32                  connID,
   pnstr8                   deviceName,
                                                    pnstr8
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier depending on the *connType* parameter.

deviceName

(IN) Points to the device name containing the necessary functions to obtain.

handle

(IN/OUT) Specifies the address where a NWPSListHandle is stored (used when calling the **NWSPdfGetNextFunction** and **NWPSPdfEndNext** function.

funcName

(OUT) Points to the next function name.

funcData

(OUT) Points to theNWPDSPdfFuncData structure, which describes the sequence of bytes comprising the function definition.

Return Values

0x0000	Successful
0xFFFF	General Error
other values	Bindery or Directory Services Errors

NCP Calls

None

See Also

NWCCOpenConnByRef, **NWCCScanConnRefs**, **NWDSCreateContextHandle**, **NWPSPdfAddDevice**, **NWPSPdfAddFunction**, **NWPSPdfDeleteFunction**, **NWPSPdfEndNext**, **NWPSPdfGetNextFunction**, **NWPSPdfReadFunction**, **NWPSPdfUpdateFunction**

NWPSPdfGetFirstMode

Finds a mode in the PRINTDEF database
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or #include <nwpsrv.h>

NWCCODE NWAPI NWPSPdfGetFirstMode (
   nuint                  connType,
   nuint32                connID,
   pnstr8                 deviceName,
   NWPSListHandle NWPTR   handle,
   pnstr8                 modeName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier (depending on the *connType* parameter).

deviceName

(IN) Points to the device name containing the mode.

handle

(IN/OUT) Specifies the address where a NWPSListHandle is stored (used in calling the **NWPSPdfGetNextMode** and **NWPSPdfEndNext** functions).

modeName

(OUT) Points to the next mode name.

Return Values

0x0000	Successful
0xFFFF	General Error

other values	Bindery or Directory Services Errors

NCP Calls

None

See Also

NWCCOpenConnByRef, **NWCCScanConnRefs**, **NWDSCreateContextHandle**, **NWPSPdfAddDevice**, **NWPSPdfAddFunction**, **NWPSPdfGetNextMode**, **NWPSPdfEndNext**, **NWPSPdfDeleteFunction**, **NWPSPdfReadFunction**, **NWPSPdfUpdateFunction**

NWPSPdfGetFirstModeFunction

Finds all the functions associated with the specified device and mode
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

NWCCODE NWAPI NWPSPdfFirstModeFunction (
   nuint                 connType,
   nuint32               connID,
   pnstr8                deviceName,
   pnstr8                modeName,
   NWPSListHandle NWPTR   handle,
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier (depending on the *connType* parameter).

deviceName

(IN) Points to the device name containing the mode.

modeName

(IN) Points to the mode name containing the necessary functions to obtain.

handle

(IN/OUT) Specifies the address where a NWPSListHandle is stored (used in calling the **NWPSPdfGetNextModeFunction** and **NWPSPdfEndNext** functions).

funcName

(OUT) Points to the next function name.

Return Values

0x0000	Successful
0xFFFF	General Error
other values	Bindery or Directory Services Errors

NCP Calls

None

See Also

NWCCOpenConnByRef, **NWCCScanConnRefs**, **NWDSCreateContextHandle**, **NWPSPdfAddDevice**, **NWPSPdfAddModeFunction**, **NWPSPdfDeleteModeFunction**, **NWPSPdfEndNext**, **NWPSPdfGetNextModeFunction**, **NWPSPdfReadModeFunction**

NWPSPdfGetImportDate

Obtains the date code of the .PDF file set by **NWPSPdfExportDevice**
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfGetImportDate (
   nuint      connType,
   nuint32    connID,
   pnstr8     fileName,
   pnuint16   year,
   pnuint16   month,
   pnuint16   day,
   pnuint16   hour,
   pnuint16   minute,
   pnuint16   second);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

fileName

(IN) Points to the name of the .PDF file.

year

(OUT) Points to the year the file was created (1992-65535).

month

(OUT) Points to the month the file was created (1-12).

day

(OUT) Points to the day the file was created (1-31).

hour

(OUT) Points to the hour the file was created (0-24, where 0=12 a.m.).

minute

(OUT) Points to the minute the file was created (0-60).

second

(OUT) Points to the second the file was created (0-60).

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL
0x7774	NWPSE_ERROR_OPENING_DB
0x7775	NWPSE_ERROR_READING_DB

Remarks

None of the fields are optional.

NCP Calls

None

See Also

NWPSPdfExportDevice, **NWPSPdfImportDevice**, **NWPSPdfSetImportDate**

NWPSPdfGetNextDevice

Finds a device in the PRINTDEF database
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

NWCCODE NWAPI NWPSPdfGetNextDevice (
   NWPSListHandle   handle,
   pnstr8           deviceName);
```

Parameters

handle

(IN) Specifies the NWPSListHandle obtained by calling the **NWPSPdfGetFirstDevice** or the **NWPSPdfGetNextDevice** function.

deviceName

(OUT) Points to the next device name.

Return Values

0x0000	Successful
0xFFFF	General Error
other values	Bindery or Directory Services Errors

NCP Calls

None

See Also

NWCCOpenConnByRef, **NWCCScanConnRefs**, **NWDSCreateContextHandle**, **NWPSPdfAddDevice**, **NWPSPdfDeleteDevice**, **NWPSPdfEndNext**, **NWPSPdfGetFirstDevice**, **NWPSPdfGetNextDevice**, **NWPSPdfUpdateDevice** **NWPSPdfReadDevice**,

NWPSPdfGetNextForm

Finds a form in the PRINTDEF database
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfGetNextForm (
   NWPSListHandle   handle,
   pnstr8           formName,
   pnuint           formNumber,
   pnuint           formLength,
   pnuint           formWidth);
```

Parameters

handle

(IN) Specifies the NWPSListHandle obtained from calling the **NWPSPdfGetFirstForm** or **NWSPdfGetNextForm** function.

formName

(OUT) Points to the next form name.

formNumber

(OUT) Points to the address of the form number (optional).

formLength

(OUT) Points to the address of the form length (optional).

formWidth

(OUT) Points to the address of the form width (optional).

Return Values

0x0000	Successful
0xFFFF	General Error
other values	Errors in Bindery or Directory Services

NCP Calls

None

See Also

NWPSPdfEndNext, **NWPSPdfExportDevice**, **NWPSPdfGetFirstForm**, **NWPSPdfGetNextForm**, **NWPSPdfImportDevice**, **NWPSPdfSetImportDate**

NWPSPdfGetNextFunction

Finds a function in the PRINTDEF database
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

NWCCODE NWAPI NWPSPdfGetNextFunction (
   NWPSListHandle          handle,
   pnstr8                  funcName,
   NWPSPdfFuncData NWPTR   funcData);
```

Parameters

handle

(IN) Specifies the NWPSListHandle obtained from the **NWSPdfGetFirstFunction** or **NWPSPdfGetNextFunction** function.

funcName

(OUT) Points to the next function name.

funcData

(OUT) Points to theNWPDSPdfFuncData structure describing the sequence of bytes comprising the function definition.

Return Values

0x0000	Successful
0xFFFFF	General Error
other values	Bindery or Directory Services Errors

NCP Calls

None

See Also

NWCCScanConnRefs, **NWCCOpenConnByRef**, **NWDSCreateContextHandle**, **NWPSPdfAddDevice**, **NWPSPdfAddFunction**, **NWPSPdfDeleteFunction**, **NWPSPdfGetFirstFunction**, **NWPSPdfGetNextFunction**, **NWPSPdfReadFunction**, **NWPSPdfUpdateFunction**

NWPSPdfGetNextMode

Finds a mode in the PRINTDEF database
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

NWCCODE NWAPI NWPSPdfGetNextMode (
   NWPSListHandle   handle,
   pnstr8           modeName);
```

Parameters

handle

(IN) Specifies the NWPSListHandle obtained from the **NWPSPdfGetFirstMode** or **NWPSPdfGetNextMode** function.

modeName

(OUT) Points to the next mode name.

Return Values

0x0000	Successful
0xFFFF	General Error
other values	Bindery or Directory Services Errors

NCP Calls

None

See Also

NWCCOpenConnByRef, **NWCCScanConnRefs**, **NWDSCreateContextHandle**, **NWPSPdfAddDevice**, **NWPSPdfAddFunction**, **NWPSPdfDeleteFunction**, **NWPSPdfGetFirstMode**, **NWPSPdfGetNextMode**, **NWPSPdfReadFunction**, **NWPSPdfUpdateFunction**

NWPSPdfGetNextModeFunction

Finds all the functions associated with the specified device and mode
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

NWCCODE NWAPI NWPSPdfGetNextModeFunction (
   NWPSListHandle NWPTR   handle,
   pnstr8                 funcName);
```

Parameters

handle

(IN/OUT) Specifies the NWPSListHandle obtained from the **NWPSPdfGetFirstModeFunction** or **NWPSPdfGetNextModeFunction** function.

funcName

(OUT) Points to the next function name.

Return Values

0x0000	Successful
0xFFFF	General Error
other values	Bindery or Directory Services Errors

NCP Calls

None

See Also

NWCCOpenConnByRef, **NWCCScanConnRefs**, **NWDSCreateContextHandle**, **NWPSPdfAddDevice**, **NWPSPdfAddModeFunction**, **NWPSPdfDeleteModeFunction**, **NWPSPdfGetFirstModeFunction**, **NWPSPdfGetNextModeFunction**,

NWPSPdfReadModeFunction

NWPSPdfGetVersion

Returns the version number of the PRINTDEF database
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfGetVersion (
   WORD           connType,
   DWORD          connID,
   DWORD NWFAR  *pdfVersion);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

pdfVersion

(OUT) Points to the storage location for the database version number. NWPS_BINDERY_SERVICE_PRE_40 always returns zero.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

The database file/stream is automatically opened and closed by **NWPSPdfGetVersion**.

If the database version is incorrect, an error occurs on all other PRINTDEF calls, except **NWPSPdfSetVersion**.

NCP Calls

None

See Also

NWCCOpenConnByName, **NWDSCreateContextHandle**

NWPSPdfImportDevice

Imports a device from a PDF file to the database
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfImportDevice (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *fileName,
   char NWFAR   *deviceName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

fileName

(IN) Points to the name of the .PDF file.

deviceName

(IN/OUT) Points to an empty string upon input. Points to the device to import upon output. Its maximum length is NWPS_DEVI_NAME_SIZE + 1.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL
0x7774	NWPSE_ERROR_OPENING_DB

0x7775	NWPSE_ERROR_READING_DB
0x89EE	OBJECT_ALREADY_EXISTS

Remarks

The *fileName* should be in one of the following forms:

```
\\<file server><volume><path><file name>[.PDF] <volume>:<path><file nam
```

If *fileName* is NULL, the file **<**device name**>**[.PDF] is created in the local directory.

deviceName changes the name of an import device in cases when it may conflict with an existing device and returns the imported device name to the caller.

When *deviceName* is NULL, use the name in the PDF file. If it is an empty string, use the name in the PDF file, but return the imported name to the user. If it is a new name, use the specified name instead of the name in the PDF file.

If OBJECT_ALREADY_EXISTS returns, pass in a string to *deviceName* not already in use.

Every PDF import file also has a date code associated with it. The date is always set when the file is created. See **NWPSPdfSetImportDate** and **NWPSPdfGetImportDate**.

NCP Calls

None

See Also

NWPSPdfExportDevice, **NWPSPdfGetImportDate**, **NWPSPdfSetImportDate**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSPdfLocateDBAndSetContext

Sets the context's DCK_NAME_CONTEXT to the Organizational Unit or Organization in the Directory where a database already exists

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95

Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfLocateDBAndSetContext (
   DWORD        contextID,
   char NWFAR  *printerOrQueueObjectName);
```

Parameters

contextID

(IN/OUT) Specifies the Directory context handle.

printerOrQueueObjectName

(IN) Points to the starting point for the search.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

NWPSPdfLocateDBAndSetContext searches the Directory toward the root using the given printer or print queue object name.

Prior to making other Printer Definition calls, **NWPSPdfLocateDBAndSetContext** is used to establish which database to refer to with respect to a particular queue or printer. This is important in applications with functions similar to NPrint and Capture. The PrintDef utility does not call **NWPSPdfLocateDBAndSetContext**.

NCP Calls

None

NWPSPdfManagedImportDevice

Reports progress of the import process by calling a designated function
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfManagedImportDevice (
   WORD                connType,
   DWORD               connID,
   char NWPTR          fileName,
   char NWPTR          deviceName,
   NWPSImportManager   importManagerFunc,
   void NWPTR          importManagerParm1);
```

Parameters

connType

(IN) Specifies the type of server/network: NWPS_DIRECTORY_SERVICE, NWPS_BINDERY_SERVICE, or NWPS_BINDERY_SERVICE_PRE_40.

connID

(IN) Specifies the NetWare server connection ID.

fileName

(IN) Points to the name of the .PDF file.

deviceName

(IN/OUT) Points to the new device name or NULL. Its maximum length is NWPS_DEVI_NAME_SIZE + 1.

im

(IN) Specifies the manager function.

importManagerParm1

(IN) Points to the user's manager context variable.

Return Values

0x0000	Successful

-1	General Error
other values	Bindery or Directory Services Errors

Remarks

NWPSPdfManagedImportDevice is an enhancement to **NWPSPdfImportDevice**, and is designed to report progress of the import process by calling a function designated by *importManagerFunc*.

Setting *importManagerFunc* and *importManagerParm1* to NULL is the equivalent to calling **NWPSPdfImportDevice**.

During the import process, *importManagerFunc* is called repeatedly to indicate the number of total calls that will be made and the current call number. The typedef shown below describes the function whose address is passed in *importManagerFunc*. This typedef is for the managed import device calls.

```
typedef NWCCODE (NWAPI *NWPSImportManager)
   (void NWFAR   *importManagerParm1,
   int            totalCallsToBeMade,
   int            currentCallCount);
```

importManagerParm1(IN) User-specific data.

totalCallsToBeMade(IN) Maximum number of calls to be made.

currentCallCount(IN) Number of this function.

importManagerParm1 is intended to give context to the caller's import manager. *totalCallsToBeMade* and *currentCallCount* are for reporting progress. For example, percent complete could be calculated as follows:

```
percentDone = 100 * currentCallCount/totalCallsToBeMade
```

If the return code is non-zero, the import aborts, and the device remnants are deleted. This could be useful in implementing a cancel feature, which is done by checking, for example, a hotkey or mouse button for interruptions. However, when *totalCallsToBeMade* and *currentCallCount* are equal, the return code is ignored. This is the caller's opportunity to do screen cleanup, etc.

NCP Calls

None

See Also

NWPSPdfImportDevice

NWPSPdfReadDevice

Reads the number of modes and functions defined for a device
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfReadDevice (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *deviceName,
   WORD NWFAR   *modeCount,
   WORD NWFAR   *funcCount);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

deviceName

(IN) Points to the name of the device to read.

modeCount

(OUT) Points to the number of modes defined for the device.

funcCount

(OUT) Points to the number of functions defined.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

NCP Calls

None

See Also

NWPSPdfAddDevice, **NWPSPdfDeleteDevice**, **NWPSPdfUpdateDevice**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSPdfReadForm

Reads a form definition from the PRINTDEF database
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfReadForm (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *formName,
   WORD NWFAR   *formNumber,
   WORD NWFAR   *formLength,
   WORD NWFAR   *formWidth);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

formName

(IN) Points to the form name for reading information.

formNumber

(OUT) Points to the space for number or NULL.

formLength

(OUT) Points to the space for length or NULL.

formWidth

(OUT) Points to the space for width or NULL.

Return Values

0x0000	Successful
-1	General Error

other values	Bindery or Directory Services Errors

NCP Calls

None

See Also

NWPSPdfAddForm, NWPSPdfDeleteForm, NWPSPdfUpdateForm, NWCCOpenConnByName, NWDSCreateContextHandle

NWPSPdfReadFunction

Reads a function definition in the PRINTDEF database
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfReadFunction (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *deviceName,
   char NWFAR   *funcName,
   WORD          funcOffset,
   WORD NWFAR   *funcSize,
   BYTE NWFAR   *funcString);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

deviceName

(IN) Points to the associated device name.

funcName

(IN) Points to the name of the function to read.

funcOffset

(IN) Specifies the starting offset or number of entries to skip past.

funcSize

(IN/OUT) Points to the buffer size of*funcString* upon input. Points to the actual number of bytes read upon output.

funcString

(OUT) Points to the buffer where the string is stored.

Return Values

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

Functions can be up to 64KB and may need to be read in segments.

funcSize should be set to the length of the *funcString* buffer. If **NWPSPdfReadFunction** is successful, *funcSize* contains the number of bytes copied to the *funcString* buffer. When no bytes are left in the database, a -1 is returned or *funcSize* is zero.

NCP Calls

None

See Also

NWPSPdfAddDevice, **NWPSPdfAddFunction**, **NWPSPdfDeleteFunction**, **NWPSPdfReadModeFunction**, **NWPSPdfUpdateFunction**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSPdfReadMode

Reads the number of functions associated with the specified mode
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfReadMode (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *deviceName,
   char NWFAR   *modeName,
   WORD NWFAR   *funcCount);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

deviceName

(IN) Points to the associated device name.

modeName

(IN) Points to the name of the mode to read.

funcCount

(OUT) Points to the number of the function defined.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

If the mode is defined and *funcCount* is not NULL, the number of functions in the mode is returned in *funcCount*.

NCP Calls

None

See Also

NWPSPdfAddDevice, **NWPSPdfAddMode**, **NWPSPdfDeleteMode**, **NWPSPdfUpdateMode**, **NWPSPdfReadFunction**, **NWPSPdfReadModeFunction**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSPdfReadModeFunction

Reads all the function strings associated with a mode into one*funcString*
NetWare Server: 2.2, 3.11, 3.12 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfReadModeFunction (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *deviceName,
   char NWFAR   *modeName,
   WORD          funcOffset,
   WORD NWFAR   *funcSize,
   BYTE NWFAR   *funcString);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

deviceName

(IN) Points to the associated device name.

modeName

(IN) Points to the mode name.

funcOffset

(IN) Specifies the number of bytes to skip.

funcSize

(IN/OUT) Points to the buffer size of*funcString* upon input. Points to the actual number of bytes read upon output.

funcString

(OUT) Points to the buffer where the string is stored.

Return Values

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

On the first call, *funcOffset* should be set to zero; and *funcSize* should be set to the size of the buffer pointed to by *funcString*. On return, *funcSize* contains the actual number of bytes copied to *funcString*. A zero is returned if there is any data after the specified *funcOffset* (start point).

If *funcSize*'s return value is less than the requested *funcSize*, there are no more bytes to get. If a call is made with *funcOffset* equal to or greater than the end of the list, an error code is returned.

Since the mode string can be up to 64K bytes, it is possible to read in segments. To read a segment, set *funcOffset* to the starting byte offset and set *funcSize* to the size of the *funcString* buffer.

If **NWPSPdfReadModeFunction** fails or runs out of memory, -1 is returned.

NCP Calls

None

See Also

NWPSPdfAddModeFunction, **NWPSPdfDeleteModeFunction**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSPdfScanDevice (obsolete 6/96)

Finds a device in PRINTDEF database but is now obsolete. Call **NWPSPdfGetFirstDevice**, **NWPSPdfGetNextDevice**, or **NWPSPdfEndNext** instead.

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT

Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

NWCCODE NWAPI NWPSPdfScanDevice
  (WORD           connType,
   DWORD          connID,
   DWORD NWFAR   *sequence,
   char NWFAR    *deviceName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE

connID

(IN) Specifies the connection or context identifier.

sequence

(IN/OUT) Points to the index. Set *sequence* to -1 before the first call; it is incremented by **NWPSPdfScanDevice (obsolete 6/96)** on every return.

deviceName

(IN/OUT) Points to the name to find if *sequence* is NULL upon input. Points to the next device name upon output. Its maximum length is NWPS_DEVI_NAME_SIZE + 1.

Return Values

0x0000	Successful
-1	General Error
other	Bindery or Directory Services Errors

values	

Remarks

Set *sequence* to -1 on the first call to find all the devices. *sequence* is reset by **NWPSPdfScanDevice (obsolete 6/96)** if a device is found. Set *sequence* to NULL and provide a *deviceName* to verify a device exists.

NCP Calls

None

See Also

NWPSPdfAddDevice, **NWPSPdfDeleteDevice**, **NWPSPdfReadDevice**, **NWPSPdfUpdateDevice**, **NWCCScanConnRefs**, **NWCCOpenConnByRef**, **NWDSCreateContextHandle**

NWPSPdfScanForm (obsolete 6/96)

Finds a form in PRINTDEF database but is now obsolete. Call **NWPSPdfGetFirstForm**, **NWPSPdfGetNextForm**, **NWPSPdfEndNext**, **NWPSPdfReadForm**, or **NWPSPdfUpdateForm** instead.

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT

Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

NWCCODE NWAPI NWPSPdfScanForm
  (WORD           connType,
   DWORD          connID,
   DWORD NWFAR   *sequence,
   char NWFAR    *formName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

sequence

(IN/OUT) Points to the index. Set *sequence* to -1 before the first call; it is incremented by **NWPSPdfScanForm (obsolete 6/96)** on every return.

formName

(IN/OUT) Points to the name to find if *sequence* is NULL upon input. Points to the next form name upon output. Its maximum length is NWPS_FORM_NAME_SIZE+1.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

If the user wants to find a specific form, the *sequence* pointer should be NULL and *formName* should be set to the desired form. If the user wants to find all the forms, *sequence* should be set to -1 on the first call, and *sequence* is updated when the function returns.

NCP Calls

None

See Also

NWCCScanConnRefs, **NWCCOpenConnByRef**, **NWDSCreateContextHandle**, **NWPSPdfAddForm**, **NWPSPdfDeleteForm**, **NWPSPdfEndNext NWPSPdfReadForm**, **NWPSPdfUpdateForm**,

NWPSPdfScanFunction (obsolete 6/96)

Finds a function in PRINTDEF database but is now obsolete. Call **NWPSPdfGetFirstFunction**, **NWPSPdfGetNextFunction**, or **NWPSPdfEndNext** instead.

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT

Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

NWCCODE NWAPI NWPSPdfScanFunction
  (WORD           connType,
   DWORD          connID,
   DWORD NWFAR   *sequence,
   char NWFAR    *deviceName,
   char NWFAR    *funcName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

deviceName

(IN) Points to the associated device name.

sequence

(IN/OUT) Points to the index. Set *sequence* to -1 before the first call; it is incremented by **NWPSPdfScanFunction (obsolete 6/96)** on every return.

funcName

(IN/OUT) Points to the name to find if *sequence* is NULL upon input. Points to the next function name upon output. Its maximum length is NWPS_FUNC_NAME_SIZE+1.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

To find all the functions associated with a device, set the *sequence* parameter to -1 on the first call. The *sequence* parameter is reset by **NWPSPdfScanFunction (obsolete 6/96)** if functions are found. To find a specific function, set the *sequence* parameter to NULL and set the *funcName* parameter to the search name.

NCP Calls

None

See Also

NWPSPdfAddDevice, **NWPSPdfAddFunction**, **NWPSPdfDeleteFunction**, **NWPSPdfReadFunction**, **NWPSPdfUpdateFunction**, **NWCCScanConnRefs**, **NWCCOpenConnByRef**, **NWDSCreateContextHandle**

NWPSPdfScanFunctionMode

Finds the modes using the specified function
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfScanFunctionMode (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *deviceName,
   char NWFAR   *modeName,
   WORD          funcOffset,
   WORD NWFAR   *funcSize,
   BYTE NWFAR   *funcString);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

deviceName

(IN) Points to the associated device name.

modeName

(IN) Points to the mode name.

funcOffset

(IN) Specifies the number of entries to skip.

funcSize

(IN/OUT) Points to the buffer size of*funcString* upon input. Points to the actual number of bytes read upon output.

funcString

(OUT) Points to the buffer where the string is stored.

Return Values

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

To find all modes associated with the specified function, set *funcSize* to -1 on the first call; it is updated when **NWPSPdfScanFunctionMode** returns.

To find a specific function associated with a mode, set *sequence* to NULL and set *funcName* to the desired function name.

NCP Calls

None

NWPSPdfScanMode (obsolete 6/96)

Finds a mode in PRINTDEF database but is now obsolete. Call **NWPSPdfGetFirstMode**, **NWPSPdfGetNextMode**, or **NWPSPdfEndNext** instead.

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT

Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

NWCCODE NWAPI NWPSPdfScanMode
  (WORD           connType,
   DWORD          connID,
   DWORD NWFAR   *sequence,
   char NWFAR    *deviceName,
   char NWFAR    *modeName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

sequence

(IN/OUT) Points to the index. Set *sequence* to -1 before the first call; it is incremented by **NWPSPdfScanMode (obsolete 6/96)** on every return.

deviceName

(IN) Points to the associated device name.

modeName

(IN/OUT) Points to the name to find if *sequence* is NULL upon input. Points to the next mode name upon output. Its maximum length is NWPS_MODE_NAME_SIZE+1.

Return Values

0x0000	Successful

-1	General Error
other values	Bindery or Directory Services Errors

Remarks

Set *sequence* to -1 on the first call to find all the modes for a device. *sequence* is reset if a mode is found. To find a specific mode, set*sequence* to NULL and set *modeName* to the desired search name.

The default mode for all devices is

```
NWPS_RESET_MODE = "(Re-initialize)"
```

NCP Calls

None

See Also

NWPSPdfAddDevice, **NWPSPdfAddMode**, **NWPSPdfDeleteMode**, **NWPSPdfReadMode**, **NWPSPdfUpdateMode**, **NWCCScanConnRefs**, **NWCCOpenConnByRef**, **NWDSCreateContextHandle**

NWPSPdfScanModeFunction (obsolete 6/96)

Finds all the functions associated with the specified mode but is now obsolete. Call **NWPSPdfGetFirstModeFunction**, **NWPSPdfGetNextModeFunction**, or **NWPSPdfEndNext** instead.

NetWare Server: 2.2, 3.11, 3.12, 4.x

Platform: DOS, OS/2, Windows 3.1, Windows NT

Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

NWCCODE NWAPI NWPSPdfScanModeFunction
  (WORD           connType,
   DWORD          connID,
   DWORD NWFAR   *sequence,
   char NWFAR    *deviceName,
   char NWFAR    *modeName,
   char NWFAR    *funcName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

sequence

(IN/OUT) Points to the index. Set *sequence* to -1 before the first call; it is incremented by **NWPSPdfScanModeFunction (obsolete 6/96)** on every return.

deviceName

(IN) Points to the associated device name.

modeName

(IN) Points to the mode name.

funcName

(IN/OUT) Points to the name to find if *sequence* is NULL upon input. Points to the next mode function name upon output. Its maximum length is NWPS_FUNC_NAME_SIZE+1.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

To find all the functions associated with a mode, set *sequence* to -1 on the first call; it resets if functions are found. To find a specific function, set *sequence* to NULL and set *funcName* to the search name.

NCP Calls

None

See Also

NWPSPdfAddDevice, **NWPSPdfAddModeFunction**, **NWPSPdfDeleteModeFunction**, **NWPSPdfReadModeFunction**, **NWCCScanConnRefs**, **NWCCOpenConnByRef**, **NWDSCreateContextHandle**

NWPSPdfSetImportDate

Changes the date and time set in the PDF file by **NWPSPdfExportDevice**
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfSetImportDate (
   nuint     connType,
   nuint32   connID,
   pnstr8    fileName,
   nuint16   year,
   nuint16   month,
   nuint16   day,
   nuint16   hour,
   nuint16   minute,
   nuint16   second);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

fileName

(IN) Points to the name of the .PDF file.

year

(IN) Specifies the year to set the date to (1992-65535).

month

(IN) Specifies the month to set the date to (1-12).

day

(IN) Specifies the day to set the date to (1-31).

hour

(IN) Specifies the hour to set the date to (0-23, where 0=12 a.m.).

minute

(IN) Specifies the minute to set the date to (0-60).

second

(IN) Specifies the second to set the date to (0-60).

Return Values

These are common return values; see 'Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL
0x7774	NWPSE_ERROR_OPENING_DB
0x7777	NWPSE_ERROR_WRITING_DB

Remarks

None of the fields are optional.

NCP Calls

None

See Also

NWPSPdfExportDevice, **NWPSPdfImportDevice**, **NWPSPdfGetImportDate**

NWPSPdfSetVersion

Sets the PDF version stored in the database
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfSetVersion (
   WORD    connType,
   DWORD   connID,
   DWORD   pdfVersion);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

pdfVersion

(IN) Specifies the database version. For NWPS_BINDERY_SERVICE_PRE_40, no change is made since the database has no place to store the version.

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x0000	NWPSE_SUCCESSFUL
0x7774	NWPSE_ERROR_OPENING_DB
0x7775	NWPSE_ERROR_READING_DB
0x7777	NWPSE_ERROR_WRITING_DB

0x7779	NWPSE_INTERNAL_ERROR

Remarks

The database file/stream is automatically opened and closed by **NWPSPdfSetVersion**.

NCP Calls

None

See Also

NWPSPdfGetVersion, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSPdfUpdateDevice

Changes a device name on PRINTDEF database
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfUpdateDevice (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *oldDeviceName,
   char NWFAR   *newDeviceName);
```

Parameters

connType

(IN) Specifies either WPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

oldDeviceName

(IN) Points to the old device name.

newDeviceName

(IN) Points to the new device name.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

NCP Calls

None

None

See Also

NWPSPdfAddDevice , NWPSPdfDeleteDevice, NWPSPdfReadDevice, NWCCOpenConnByName, NWDSCreateContextHandle

NWPSPdfUpdateForm

Changes a form name or description in PRINTDEF database
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfUpdateForm (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *oldFormName,
   char NWFAR   *newFormName,
   WORD          formNumber,
   WORD          formLength,
   WORD          formWidth);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

oldFormName

(IN) Points to the old form name to change.

newFormName

(IN) Points to the new form name or NULL.

formNumber

(IN) Specifies the new form number or -1.

formLength

(IN) Specifies the new form length or -1.

formWidth

(IN) Specifies the new form width or -1.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

If *newFormName* is NULL, the form name is not changed, but the parameters may still be changed.

If any parameter is -1, the value is not changed.

NCP Calls

None

See Also

NWPSPdfAddForm, **NWPSPdfDeleteForm**, **NWPSPdfReadForm**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSPdfUpdateFunction

Changes a function name or string
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfUpdateFunction (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *deviceName,
   char NWFAR   *oldFuncName,
   char NWFAR   *newFuncName,
   WORD          funcSize,
   BYTE NWFAR   *funcString);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

deviceName

(IN) Points to the associated device name.

oldFuncName

(IN) Points to the function name to modify.

newFuncName

(IN) Points to the new function name or NULL.

funcSize

(IN) Specifies the length of the new string or -1.

funcString

(IN) Points to the new string or NULL.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

To change the function name, set *newFuncName* to a new name. To leave the function name the same, set *newFuncName* to NULL.

To change the function string, set *funcSize* to the number of bytes in *funcString* and set *funcString* to point to the new string. To leave the string the same, set *funcSize* to -1.

NCP Calls

None

See Also

NWPSPdfAddDevice, **NWPSPdfAddFunction**, **NWPSPdfDeleteFunction**, **NWPSPdfReadFunction**, **NWPSPdfUpdateFunction**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSPdfUpdateMode

Changes the name of a mode in PRINTDEF database
NetWare Server: 2.2, 3.11, 3.12, 4.x
Platform: DOS, OS/2, Windows 3.1, Windows NT, Windows95
Service: Print Server

Syntax

```
#include <nwps_pdf.h>
or
#include <nwpsrv.h>

N_EXTERN_LIBRARY(NWCCODE) NWPSPdfUpdateMode (
   WORD          connType,
   DWORD         connID,
   char NWFAR   *deviceName,
   char NWFAR   *oldModeName,
   char NWFAR   *newModeName);
```

Parameters

connType

(IN) Specifies either NWPS_BINDERY_SERVICE, NWPS_BINDERY_SERVICE_PRE_40, or NWPS_DIRECTORY_SERVICE.

connID

(IN) Specifies the connection or context identifier.

deviceName

(IN) Points to the associated device name.

oldModeName

(IN) Points to the former mode name.

newModeName

(IN) Points to the new mode name.

Return Values

0x0000	Successful
-1	General Error
other values	Bindery or Directory Services Errors

Remarks

All mode function relations are left intact.

NCP Calls

None

See Also

NWPSPdfAddDevice, **NWPSPdfAddMode**, **NWPSPdfDeleteMode**, **NWPSPdfReadMode**, **NWCCOpenConnByName**, **NWDSCreateContextHandle**

NWPSRegisterLibraryClient

Enables the use of NWPS functions in NLM applications
Local Servers: N/A
Remote Servers: N/A
NetWare Server: 3.x, 4.x
Platform: NLM
Service: Print Server

Syntax

```
#include <nwps_nlm.h>
#include <nwps_err.h>

int NWPSRegisterLibraryClient (
   void);
```

Return Values

These are common return values; see "Return Values" in the *NetWare SDK Getting Started* for more information.

0x7 7B0	NWPSE_UNREGISTERED_THREAD
0x7 7B1	NWPSE_UNREGISTERED_NLM

Remarks

NWPSRegisterLibraryClient must be called before any other Print Server Services functions are called.

NWPSRV3X.NLM must be used for **NWPSDeRegisterLibraryClient** to be called in a 3.x environment.

NCP Calls

None

Print Service Group

Print Server: Structures

NWPS_Aio

Defines configuration information specific to AIO printers
Service: Print Server
Defined In: nwps_cfg.h

Structure

```
typedef struct NWPS_Aio_t
{
   nuint16   reserved1;
   nuint16   startFlag;
   nuint16   hardwareType;
   nuint8    boardNumber;
   nuint8    portNumber;
   nuint8    useXonXoff;
   nuint8    aioMgr;
   nuint16   baudRate;
   nuint16   dataBits;
   nuint16   stopBits;
   nuint16   parity;
} NWPS_Aio;
```

Fields

reserved1

Is reserved for future use (AIO).

startFlag

(IN) Specifies when to start: 1 = NWPS_AUTO_START; 0 = NWPS_USER_START

hardwareType

(IN) Specifies the hardware type.

boardNumber

(IN) Specifies the board number.

portNumber

(IN) Specifies the COM *n* port number (serial). LPT *n* port number (parallel).

useXonXoff

(IN) Specifies on/off; use X=On/X=Off.

aioMgr

(IN) Specifies the AIO interface.

baudRate

(IN) Specifies the baud rate.

dataBits

(IN) Specifies the data bits.

stopBits

(IN) Specifies the stop bits.

parity

(IN) Specifies the parity type.

NWPS_AppleTalk

Defines configuration information specific to AppleTalk printers
Service: Print Server
Defined In: nwps_cfg.h

Structure

```
typedef struct NWPS_AppleTalk_t
{
   char      netPrinterName[NWPS_APPLE_NAME_SIZE + 2];
   char      netPrinterType[NWPS_APPLE_TYPE_SIZE + 2];
   char      netPrinterZone[NWPS_APPLE_ZONE_SIZE + 2];
   nuint16   hideFlag;
   nuint16   errorFlag;
} NWPS_AppleTalk;
```

Fields

netPrinterName

(IN) Specifies the AppleTalk network printer name.

netPrinterType

(IN) Specifies the AppleTalk network printer type.

netPrinterZone

(IN) Specifies the AppleTalk network printer zone.

hideFlag

(IN) Specifies whether to hide the flags: TRUE = Hide printer.

errorFlag

(IN) Specifies the error: TRUE = Print error banner.

NWPS_Job_Rec

Service: Print Server
Defined In: nwps_cfg.h

Structure

```
typedef struct
{
   nuint32   printJobFlag;
   nuint16   copies;
   nuint16   timeOutCount;
   nuint8    tabSize;
   nuint8    localPrinter;
   char   formName[NWPS_FORM_NAME_SIZE + 2];
   char   name[NWPS_BANNER_NAME_SIZE + 2];
   char   bannerName[NWPS_BANNER_FILE_SIZE + 2];
   char   device[NWPS_DEVI_NAME_SIZE + 2];
   char   mode[NWPS_MODE_NAME_SIZE + 2];
   union
   {
      struct {
         char   fileServer[NWPS_BIND_NAME_SIZE + 2];
         char   printQueue[NWPS_BIND_NAME_SIZE + 2];
         char   printServer[NWPS_BIND_NAME_SIZE + 2];
      } nonDS;
   char   DSObjectName[NWPS_MAX_NAME_SIZE];
   } u;
   nuint8    reserved[390];
} NWPS_Job_Rec;
```

Fields

printJobFlag

(IN) Specifies the following bit definition:

```
0 = File type: 1=Text; 0=Byte stream
1 = Suppress formfeed: 0=No;1=Yes
2 = Notify when done: 0=No;1=Yes
3 = Print banner: 0=No;1=Yes
4 = Auto endcap: 0=No;1=Yes
5 = Enable timeout: 0=No;1=Yes
8-6 = Environment: 000=Bindery 001=Directory Services
11-9 = Destination: 000=Queue_Name 001=Printer_Name
31-12 = Unused
```

copies

(IN) Specifies the number of copies (from 1 to 65,000 characters long).

timeOutCount

(IN) Specifies how long to wait before printing (from 1 to 1,000 characters long).

tabSize

(IN) Specifies the tab size which is a value between 1 and 18 inclusive (default is 0x08).

localPrinter

(IN) Specifies the local printer: 0=Lpt1; 1=Lpt2; 2=Lpt3.

formName

(IN) Specifies the name of the form a user must mount in the printer to print files captured to the LPT device. If the form currently mounted in the printer differs from the form name returned in this field, the NetWare server console displays a message instructing the console operator to mount the correct form

name

(IN) Specifies the name (from 1 to 12 characters long).

bannerName

(IN) Specifies the banner name (from 1 to 12 characters long).

device

(IN) Specifies the device name (from 1 to 32 characters long).

mode

(IN) Specifies the mode (from 1 to 32 characters long).

fileServer

(IN) Specifies the file server (from 2 to 48 characters long).

printQueue

(IN) Specifies the print queue (from 1 to 48 characters long).

printServer

(IN) Specifies the print server (from 1 to 48 characters long).

DSObjectName

(IN) Specifies the object name.

reserved

(IN) Specifies up to 1024 total additional bytes.

NWPS_NInfo

Service: Print Server
Defined In: nwps_com.h

Structure

```
typedef struct
{
   nuint16   printerType;
   nuint16   useInterrupts;
   nuint16   irqNumber;
   nuint16   numBlocks;
   nuint16   useXonXoff;
   nuint16   baudRate;
   nuint16   dataBits;
   nuint16   stopBits;
   nuint16   parity;
   nuint16   socket;
} NWPS_NInfo
```

Fields

printerType

(IN) Specifies the type of network printer.

useInterrupts

(IN) Specifies if interrupts should be used.

irqNumber

(IN) Specifies the IRQ number for printer.

numBlocks

(IN) Specifies the number of blocks in buffer.

useXonXoff

(IN) Specifies whether to use Xon/Xoff: YES=1, NO=0.

baudRate

(IN) Specifies the baud rate (serial):

0 NWPS_BAUD_RATE_0300
1 NWPS_BAUD_RATE_0600
2 NWPS_BAUD_RATE_1200
3 NWPS_BAUD_RATE_2400
4 NWPS_BAUD_RATE_4800
5 NWPS_BAUD_RATE_9600
6 NWPS_BAUD_RATE_19200
7 NWPS_BAUD_RATE_38400

dataBits

(IN) Specifies the number of data bits (serial):

5 NWPS_DATA_BITS_5
6 NWPS_DATA_BITS_6
7 NWPS_DATA_BITS_7
8 NWPS_DATA_BITS_8

stopBits

(IN) Specifies the number of stop bits (serial):

0 NWPS_STOP_BITS_1
1 NWPS_STOP_BITS_1_5
2 NWPS_STOP_BITS_2

parity

(IN) Specifies the parity type (serial):

0 NWPS_PARITY_NONE
1 NWPS_PARITY_EVEN
2 NWPS_PARITY_ODD

socket

(IN) Specifies the socket number for network printers.

NWPS_NStatus

Service: Print Server
Defined In: nwps_com.h

Structure

```
typedef struct
{
   nuint8    printerNumber;
   nuint8    needBlocks;
   nuint8    finishedBlocks;
   nuint8    numBlocks;
   nuint8    status;
   nuint8    insideBand;
} NWPS_NStatus
```

Fields

printerNumber

(IN) Specifies network printer number.

needBlocks

(IN) Specifies number of blocks needed to fill buffers.

finishedBlocks

(IN) Specifies number of blocks printed.

status

(IN) Specifies whether the printer is online, offline, or out-of-paper.

insideBand

(IN) Specifies whether the network printer is in sideband mode.

NWPS_Parallel

Defines configuration information specific to parallel printers
Service: Print Server
Defined In: nwps_cfg.h

Structure

```
typedef struct NWPS_Parallel_t
}
   nuint16   portNumber;
   nuint16   startFlag;
   nuint16   useInterrupts
   nuint16   irqNumber;
} NWPS_Parallel;
```

Fields

portNumber

(IN) Specifies the LPT *n* port number (parallel).

startFlag

(IN) Specifies when to start the printer: 1 = NWPS_AUTO_START; 0 = NWPS_USER_START

useInterrupts

(IN) Specifies whether to use interrupts: TRUE = Use IRQ driver.

irqNumber

(IN) Specifies the IRQ number for printer.

NWPS_PConfig

Returns default configuration information for specific printer types
Service: Print Server
Defined In: nwps_cfg.h

Structure

```
typedef struct {
   nuint16      printerType;
   nuint16      currentForm;
   nuint16      bufferSize;
   nuint16      serviceMode;
   nuint16      pollTime;
   nuint16      bannerType;
   nuint32      length;
   char         driverName[9];
   char         reserved_for_future[23];
   union {
      NWPS_Serial      ser;
      NWPS_Parallel    par;
      NWPS_Aio         aio;
      NWPS_AppleTalk   apl;
      NWPS_Unix        unx;
      nuint8           oth[NWPS_OTHER_SIZE];
    } type;
} NWPS_PConfig;
```

Fields

printerType

(IN) Specifies possible printer types listed below:

```
0 = NWPS_P_OTHER
1 = NWPS_P_PA
2 = NWPS_P_SER
3 = NWPS_P_XNP
4 = NWPS_P_APPLE
5 = NWPS_P_UNIX
6 = NWPS_P_AIO
1 = NWPS_P_ELSEWHERE
```

currentForm

(IN) Specifies the currently mounted form.

bufferSize

(IN) Specifies the buffer size in kilobytes.

serviceMode

(IN) Specifies the queue service mode.

pollTime

(IN) Specifies the queue poll time.

bannerType

(IN) Specifies the type of banner page: FALSE = text banner page; TRUE = postscript banner page

length

(IN) Specifies the length of other data.

driverName

(IN) Specifies the NLM filename (without the extension) to load (up to 8 characters plus a zero).

reserved_for_future

(IN) Specifies the even number of nuint32s.

ser

(IN) Specifies the NWPS_Serial structure.

par

(IN) Specifies the NWPS_Parallel structure.

aio

(IN) Specifies the NWPS_Aio structure.

apl

(IN) Specifies the NWPS_AppleTalk structure.

unx

(IN) Specifies the NWPS_Unix structure.

oth

(IN) Specifies the other size.

NWPS_PSInfo

Defines print server information returned by **NWPSComGetPrintServerInfo**

Service: Print Server

Defined In: nwps_com.h

Structure

```
typedef struct
{
   nuint8   status;
   nuint8   numPrinters;
   nuint8   numModes;
   nuint8   majorVersion;
   nuint8   minorVersion;
   nuint8   revision;
   nuint8   serialNumber[4];
   nuint8   serverType;
   nuint8   nameServiceMode
   nuint8   futureUse[8];
} NWPS_PSInfo;
```

Fields

status

(IN) Specifies the print server status code which are:

```
0 NWPS_RUNNING
1 NWPS_GOING_DOWN
2 NWPS_DOWN
3 NWPS_INITIALIZING
```

numPrinters

(IN) Specifies the number of attached printers. The range is from 0 to 15.

numModes

(IN) Specifies the number of queue service modes supported by the print server. Possible queue service modes include the following values:

```
0  NWPS_QUEUE_ONLY
1  NWPS_QUEUE_BEFORE_FORM
2  NWPS_FORM_ONLY
3  NWPS_FORM_BEFORE_QUEUE
```

majorVersion

(IN) Specifies the print server protocol, major version.

minorVersion

(IN) Specifies the print server protocol, minor version.

revision

(IN) Specifies the print server protocol, revision.

serialNumber

(IN) Specifies the serial number in BCD.

serverType

(IN) Specifies the print server types which are:

```
0  NWPS_TYPE_UNKNOWN
1  NWPS_TYPE_EXE
2  NWPS_TYPE_NLM
3  NWPS_TYPE_SERVER_VAP
4  NWPS_TYPE_BRIDGE_VAP
5  NWPS_TYPE_UNIX
```

nameServiceMode

(IN) Specifies the service mode: NWPS_BINDERY_SERVICE or NWPS_DIRECTORY_SERVICE.

futureUse

Is reserved for future use.

NWPS_Serial

Defines configuration information specific to serial printers
Service: Print Server
Defined In: nwps_cfg.h

Structure

```
typedef struct NWPS_Serial_t
{
   nuint16   portNumber;
   nuint16   startFlag;
   nuint16   useInterrupts;
   nuint16   irqNumber;
   nuint16   useXonXoff;
   nuint16   baudRate;
   nuint16   dataBits;
   nuint16   stopBits;
   nuint16   parity;
} NWPS_Serial;
```

Fields

portNumber

(IN) Specifies the COM *n* port number (serial).

startFlag

(IN) Specifies when to start the printer: 1 = NWPS_AUTO_START.

useInterrupts

(IN) Specifies whether to use interrupts: TRUE = Use IRQ driver.

irqNumber

(IN) Specifies the IRQ number for printer.

useXonXoff

(IN) Specifies whether to use on/off: X=On/X=Off.

baudRate

(IN) Specifies the baud rate (serial):

0 NWPS_BAUD_RATE_0300
1 NWPS_BAUD_RATE_0600
2 NWPS_BAUD_RATE_1200
3 NWPS_BAUD_RATE_2400
4 NWPS_BAUD_RATE_4800
5 NWPS_BAUD_RATE_9600
6 NWPS_BAUD_RATE_19200
7 NWPS_BAUD_RATE_38400

dataBits

(IN) Specifies the data bits (serial):

5 NWPS_DATA_BITS_5
6 NWPS_DATA_BITS_6
7 NWPS_DATA_BITS_7
8 NWPS_DATA_BITS_8

stopBits

(IN) Specifies the stop bits (serial):

0 NWPS_STOP_BITS_1
1 NWPS_STOP_BITS_1_5
2 NWPS_STOP_BITS_2

parity

(IN) Specifies the parity type (serial):

0 NWPS_PARITY_NONE
1 NWPS_PARITY_EVEN
2 NWPS_PARITY_ODD

NWPS_Typed_Name

Holds information for the the Operator, User, Owner, and Notify attributes
Service: Print Server
Defined In: nwps_cfg.h

Structure

```
typedef struct
{
   nuint16   objectType;
   nptr      tName;
} NWPS_Typed_Name;
```

Fields

objectType

(IN) Specifies the object type.

tName

(IN) Points to the Typed_Name_T structure.

Remarks

The *level* field of the Typed_Name_T structure contains the printer number which can be passed to the *printerID* parameter in NWPSComxxx functions.

NWPS_Unix

Defines configuration information specific to Unix Printers
Service: Print Server
Defined In: nwps_cfg.h

Structure

```
typedef struct NWPS_Unix_t
{
   char   hostName[NWPS_UNIX_HOST_SIZE + 1];
   char   hostPrinter[NWPS_UNIX_PRNT_SIZE + 1];
} NWPS_Unix;
```

Fields

hostName

(IN) Specifies the name of the Unix host.

hostPrinter

(IN) Specifies the Unix printer name.