6

Path Functions

The functions contained in this chapter provide support for MAC and other[3] types of path strings.

NetWare Macintosh path strings have the following format:

"volume::dir:dir:file"

DOS path strings have the following format:

Server\volume:dir\dir\file

And OS/2, FTAM, and NFS path strings have the following format:

root/dir/file

The following data structure is used in this chapter:

```
typedef struct
{
   UINT16  size;
   CHAR    string[1];
} STRING_BUFFER;
```

where *size* is the size of the *path* buffer in bytes and *string* points to the path.

[3] DOS, FTAM, OS/2, and NFS

STRING
NWSMStripPathChild -- Unsupported

(UINT32 nameSpaceType,
  STRING path,
  STRING child,
  size_t maxChildLength);

Parameters

nameSpaceType	(INPUT) Passes the child's name space type.
path	(INPUT) Passes the path information.
child	(OUTPUT) Passes a buffer and returns the child's name.
maxChildLength	(INPUT) Passes the maximum size of *child*.

Completion Codes

NULL	No child and no separator found
Non-null	A child or separator was found

Remarks

NWSMStripPathChild removes the child from *path* and transfers it to *child*, then overwrites the first character of the child in *path* with a NULL. To restore the child to *path*, copy the first character of *child* into the memory location returned by the function. If there is no child, *child equals '\0'. This function works only for DOS and MAC under NetWare 3.xx and beyond; and for NFS, OS/2, and FTAM under NetWare 3.11 and beyond.

Caution: This is an unsupported function. It may not appear in future releases of the SMS utilities library.

Example

```
#include  <smslib.h>

STRING  resultString;
UINT32  nameSpaceType;
STRING  path;
STRING  child;
size_t  maxChildLength;

resultString = NWSMStripPathChild(nameSpaceType,path,child,maxChildLength);
```

STRING
NWSMGetPathChild -- Unsupported

(UINT32 nameSpaceType,
  STRING path,
  STRING *child);

Parameters

nameSpaceType	(INPUT) Passes the child's name space type.
path	(INPUT) Passes the path information
child	(OUTPUT) Returns a pointer to the child.

Completion Codes

NULL	No child found
Non-null	A child was found

Remarks

NWSMGetPathChild returns a pointer to the child, but if no child is found, a NULL is returned. This function works only for DOS and MAC under NetWare 3.xx and beyond; and for NFS, OS/2, and FTAM under NetWare 3.11 and beyond.

Caution: This is an unsupported function. It may not appear in future releases of the SMS utilities library.

Example

```
#include  <smslib.h>

STRING  resultString;
UINT32  nameSpaceType;
STRING  path;
STRING  child;

resultString = NWSMGetPathChild(nameSpaceType,path,&child);
```

NWBOOLEAN

NWSMPathIsNotFullyQualified -- Unsupported

(UINT32 nameSpaceType,
STRING path);

Parameters

nameSpaceType	(INPUT) Passes the name space type.
path	(INPUT) Passes the path information.

Completion Codes

TRUE	The path is not fully qualified
FALSE	

Remarks

A fully qualified path's format is:

vol:path

and not a relative path (e.g., a path relative to the current directory). This function works only for DOS and MAC under NetWare 3.xx and beyond; and for NFS, OS/2, and FTAM under NetWare 3.11 and beyond.

Caution: This is an unsupported function. It may not appear in future releases of the SMS utilities library.

Example

```
#include  <smslib.h>

NWBOOLEAN  notQualified;
UINT32     nameSpaceType;
STRING     path;

notQualified = NWSMPathIsNotFullyQualified(nameSpaceType,path);
```

NWBOOLEAN

NWSMLegalDOSName -- Unsupported

(STRING name);

Parameters

name	(INPUT) Passes the terminal node name only. It must not be NULL (e.g., *name* = NULL or **name* = '\0').

Completion Codes

TRUE	
FALSE	*name* is ".", "..", or does not meet the proper DOS format.

Remarks

NWSMLegalDOSName checks for valid name length and extension length. This function works only for the DOS or NetWare name spaces (i.e., volume:path).

> **Caution**: This is an unsupported function. It may not appear in future releases of the SMS utilities library.

Example

```
#include <smslib.h>

NWBOOLEAN  legalName;
STRING     name;

legalName = NWSMLegalDOSName(name);
```

STRING
NWSMGetVolume -- Unsupported

```
( STRING *ptr,
  UINT32 nameSpaceType,
  STRING volume);
```

Parameters

ptr	(INPUT) Passes the path information. On return *ptr* points to the next path element.
nameSpaceType	(INPUT) Passes the data set's name space type.
volume	(OUTPUT) Passes a buffer (17 bytes) and returns the volume name.

Completion Codes

NULL	No volume name was found or its length is greater than 16 characters.
Non-null	A volume name was found (the returned pointer points to volume name)

Remarks

NWSMGetVolume returns the volume name for a name space[4]. The function returns NULL if there is no volume name or the volume name is greater than 16 characters. Otherwise, the function returns a pointer to *volume*.

This function works only for DOS and MAC under NetWare 3.xx and beyond; and for NFS, OS/2, and FTAM under NetWare 3.11 and beyond.

> **Caution**: This is an unsupported function. It may not appear in future releases of the SMS utilities library.

Example

```
#include  <smslib.h>

STRING resultString;
STRING ptr;
UINT32 nameSpaceType;
STRING volume;

resultString = NWSMGetVolume(&ptr,nameSpaceType,volume);
```

[4] DOS, MAC, FTAM, OS/2 and NFS.

CHAR
NWSMStripEndSeparator -- Unsupported

(UINT32 nameSpaceType,
  STRING path,
  CHAR *separatorPos);

Parameters

nameSpaceType	(INPUT) Passes the name space type.
path	(INPUT) Passes the path.
separatorPos	(OUTPUT) *separatorPos* points to where the end separator was.

Completion Codes

NULL	No end separator was found.
Non-null	An end separator was (the returned pointer points to end separator).

Remarks

NWSMStripEndSeparator removes the end separator and returns the separator character at the end of the path. The last character in path must be a separator: (slash). Also, this function works only for DOS and MAC under NetWare 3.xx and beyond; and for NFS, OS/2, and FTAM under NetWare 3.11 and beyond. This function works for the following name spaces:

NFSNameSpace
FTAMNameSpace
OS2NameSpace
MACNameSpace
DOSNameSpace

Caution: This is an unsupported function. It may not appear in future releases of the SMS utilities library.

Example

```
#include  <smslib.h>

CHAR   separator;
UINT32 nameSpaceType;
STRING path;
STRING separatorPos;

separator = NWSMStripEndSeparator(nameSpaceType,path,&separatorPos);
```

STRING
NWSMFixDOSPath -- Unsupported

(STRING path,
  STRING newPath);

Parameters

path	(INPUT/OUTPUT) Passes a fully qualified path. *path* may contain the fixed up string (see "Remarks" for more information).
newPath	(OUTPUT) May contain the fixed up string (see "Remarks" for more information). *newPath* must be as large as *path*.

Completion Codes

NULL	Cannot convert *path*.
Non-null	The conversion was successful (the returned pointer points to the converted string).

Remarks

NWSMFixDOSPath fixes a fully qualified path. It capitalizes the string, converts all directory separators to a / (forwardslash) and converts ":/" or ":\\" to a ":". For example, the following string is converted as shown:

"volume:\\tools\\tcpp" → "VOLUME:TOOLS/TCPP"

The converted string may be in *path* or *newPath*. The function always returns a pointer to the converted string. This function works only for the DOS or NetWare name spaces (i.e., volume:path).

> **Caution**: This is an unsupported function. It may not appear in future releases of the SMS utilities library.

Example

```
#include  <smslib.h>

STRING  resutlString;
STRING  path;
STRING  newPath;

resultString = NWSMFixDOSPath(path,newPath);
```

STRING
NWSMCatString

(STRING_BUFFER **dest,
  void *source,
  INT16 srcLen);

Parameters

dest	(OUTPUT) Passes the address of a pointer to a STRING_BUFFER structure. If *dest is NULL, memory is allocated for the string.
source	(INPUT) Passes the string to be concatenated.
srcLen	(INPUT) Passes the length of *source*. If set to -1, the length of *source* is calculated.

Completion Codes

NULL	Cannot concatenate the strings.
Non-null	The strings are concatenated.

Remarks

NWSMCatString concatenates source to dest->string and returns a pointer to dest->string if successful; otherwise a NULL is returned. To free the string, call **NWSMFreeString**.

Example

```
#include  <smslib.h>

STRING     resultString;
STRING_BUFFER *dest = NULL;
void      *source;
INT16      srcLen;

resultString = NWSMCatString(&dest, source, srcLen);
```

See Also

NWSMFreeString

STRING
NWSMStr

```
( UINT8 n,
  void *dest,
  void *src1,
  void *src2,
  ...);
```

Parameters

n	(INPUT) Passes the number of strings to be concatenated into *dest*.
dest	(OUTPUT) Returns the concatenated strings. The caller must ensure that *dest* is large enough to hold all the strings and is not a NULL pointer.
src1	(INPUT) Passes the first string to be concatenated.
src2	(INPUT) Passes the second string to be concatenated.
...	(INPUT) Passes other strings to be concatenated.

Completion Codes

Non-Zero	Success
0x0	Failure

Remarks

NWSMStr concatenates *src1*, *src2*, etc., into *dest* and returns a pointer to the concatenated string. It assumes that *dest* can contain the concatenated string.

Example

```
#include  <smslib.h>

CHAR  dest[100];
UINT8   n = 2;
void   *src1;
void   *src2;
. . .
resultString = NWSMStr(n, dest, src1, src2);
```

See Also

STRING
NWSMCopyString

(STRING_BUFFER **dest,
  void *src,
  INT16 srcLen);

Parameters

dest	(OUTPUT) Passes the address of a pointer to a STRING_BUFFER structure. If *dest is NULL, memory is allocated for the string.
src	(INPUT) Passes the string.
srcLen	(INPUT) Passes the length of *src*. If set to -1 the length of *src* is calculated.

Completion Codes

NULL	The string cannot be copied.
Non-null	The string was copied (the returned pointer points to *dest*).

Remarks

NWSMCopyString copies src to dest->string and returns a pointer to dest->string if successful; otherwise a NULL is returned. To free the string, call **NWSMFreeString**.

Example

```
#include  <smslib.h>

STRING     resultString;
STRING_BUFFER *dest = NULL;
void      *src;
INT16      srcLen;

resultString = NWSMCopyString(&dest,src,srcLen);
```

See Also

NWSMFreeString

STRING
NWSMCatStrings

(UINT8 numStrings,
STRING_BUFFER **dest,
void *src1,
void *src2,
...);

Parameters

numStrings	(INPUT) Passes the number of strings to concatenate.
dest	(OUTPUT) Passes the address of a pointer to a STRING_BUFFER structure. If *dest is NULL, memory is allocated for the string.
src1	(INPUT) Passes the first string to concatenate.
src2	(INPUT) Passes the second string to concatenate.
...	(INPUT) Passes other strings to concatenate.

Completion Codes

NULL	Cannot concatenate the strings.
Non-null	The strings are concatenated (the returned pointer points to *dest*).

Remarks

NWSMCatStrings concatenates strings into dest->string and returns a pointer to dest->string. To free the string, call **NWSMFreeString**.

Example

```
#include <smslib.h>

STRING     resultString;
UINT8      numStrings;
STRING_BUFFER *dest = NULL;
void      *src1;
void      *src2;

resultString = NWSMCatStrings(numStrings,&dest,src1,src2);
```

See Also

NWSMFreeString

VOID

NWSMFreeString

(STRING_BUFFER **string);

Parameters

string	(INPUT) Passes the address of a pointer to a STRING_BUFFER structure.

Completion Codes

None	

Remarks

NWSMFreeString frees the memory allocated to *string*.

Example

```
#include  <smslib.h>

STRING_BUFFER *string;

NWSMFreeString(&string);
```

STRING_BUFFER *

NWSMAllocString

(STRING_BUFFER **string,
INT16 size);

Parameters

string	(OUTPUT) Passes the address of a pointer to a STRING_BUFFER structure. If *string is NULL, a new structure is allocated, otherwise **NWSMAllocString** assumes that it is a valid pointer and reallocates more memory for the structure.
size	(INPUT) Passes the desire size of the new or reallocated structure. If *size* is less than zero, a standard size is allocated to the structure; otherwise, the standard size is added to the existing structure size.

Completion Codes

NULL	Cannot reallocate memory.
Non-null	Reallocation is successful (the returned pointer points to *string*)

Remarks

NWSMAllocString allocates a new structure or gives the existing structure more memory. The returned pointer is valid only if **string* is set to NULL. To free the string, call **NWSMFreeString**.

Example

```
#include  <smslib.h>

STRING_BUFFER *newString, *string;
INT16      size;

newString = NWSMAllocString(&string, size);
```

See Also

NWSMFreeString

NWBOOLEAN

NWSMIsWild

(STRING string);

Parameters

string	(INPUT) Passes the path information.

Completion Codes

TRUE	*string* contains a wildcard character.
FALSE	*string* does not contain a wildcard character.

Remarks

NWSMIsWild returns TRUE if one of the following characters is found: * (asterisks), ? (question mark), and . (period--the parity bit must be set for this character). For more information about paths and wildcard patterns, see the *Professional Development Series, Network C for DOS, NetWare System Interface Technical Overview* document (part number 100-000569-002).

Example

```
#include  <smslib.h>

NWBOOLEAN  isWild;
STRING     string;

isWild = NWSMIsWild(string);
```

NWBOOLEAN

NWSMWildMatch

(STRING pattern,
  STRING string);

Parameters

pattern	(INPUT) Passes the wildcard pattern to be searched for.
string	(INPUT) Passes the string to search.

Completion Codes

TRUE	*string* meets the wildcard's specifications.
FALSE	*string* does not have the specified wildcard or does not meet the wildcard's format.

Remarks

NWSMWildMatch checks to see if *string* matches the wild card's in *pattern*.

Example

```
#include  <smslib.h>

CCODE      isWildMatch;
STRING     pattern;
STRING     string;

isWildMatch = NWSMWildMatch(pattern,string);
```

See Also

NWSMIsWild