Novell Developer Kit

www.novell.com

INTERNATIONALIZATION

October 5, 2005

Novell®

Legal Notices

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.
www.novell.com

Online Documentation: To access the online documentation for this and other Novell developer products, and to get updates, see developer.novell.com/ndk. To access online documentation for Novell products, see www.novell.com/documentation.

Novell Trademarks

AppNotes is a registered trademark of Novell, Inc.
AppTester is a registered trademark of Novell, Inc. in the United States.
ASM is a trademark of Novell, Inc.
BorderManager is a registered trademark of Novell, Inc.
BrainShare is a registered service mark of Novell, Inc., in the United States and other countries.
C3PO is a trademark of Novell, Inc.
Certified Novell Engineer is a service mark of Novell, Inc.
Client32 is a trademark of Novell, Inc.
CNE is a registered service mark of Novell, Inc.
ConsoleOne is a registered trademark of Novell, Inc.
Controlled Access Printer is a trademark of Novell, Inc.
Custom 3rd-Party Object is a trademark of Novell, Inc.
DeveloperNet is a registered trademark of Novell, Inc., in the United States and other countries.
DirXML is a registered trademark of Novell, Inc.
eDirectory is a trademark of Novell, Inc.
Excelerator is a trademark of Novell, Inc.
exteNd is a trademark of Novell, Inc.
exteNd Director is a trademark of Novell, Inc.
exteNd Workbench is a trademark of Novell, Inc.
FAN-OUT FAILOVER is a trademark of Novell, Inc.
GroupWise is a registered trademark of Novell, Inc., in the United States and other countries.
Hardware Specific Module is a trademark of Novell, Inc.
Hot Fix is a trademark of Novell, Inc.
iChain is a registered trademark of Novell, Inc.
Internetwork Packet Exchange is a trademark of Novell, Inc.
IPX is a trademark of Novell, Inc.
IPX/SPX is a trademark of Novell, Inc.
jBroker is a trademark of Novell, Inc.
Link Support Layer is a trademark of Novell, Inc.
LSL is a trademark of Novell, Inc.
ManageWise is a registered trademark of Novell, Inc., in the United States and other countries.
Mirrored Server Link is a trademark of Novell, Inc.
Mono is a registered trademark of Novell, Inc.
MSL is a trademark of Novell, Inc.
My World is a registered trademark of Novell, Inc., in the United States.
NCP is a trademark of Novell, Inc.
NDPS is a registered trademark of Novell, Inc.
NDS is a registered trademark of Novell, Inc., in the United States and other countries.
NDS Manager is a trademark of Novell, Inc.
NE2000 is a trademark of Novell, Inc.
NetMail is a registered trademark of Novell, Inc.
NetWare is a registered trademark of Novell, Inc., in the United States and other countries.
NetWare/IP is a trademark of Novell, Inc.
NetWare Core Protocol is a trademark of Novell, Inc.
NetWare Loadable Module is a trademark of Novell, Inc.

NetWare Management Portal is a trademark of Novell, Inc.

NetWare Name Service is a trademark of Novell, Inc.

NetWare Peripheral Architecture is a trademark of Novell, Inc.

NetWare Requester is a trademark of Novell, Inc.

NetWare SFT and NetWare SFT III are trademarks of Novell, Inc.

NetWare SQL is a trademark of Novell, Inc.

NetWire is a registered service mark of Novell, Inc., in the United States and other countries.

NLM is a trademark of Novell, Inc.

NMAS is a trademark of Novell, Inc.

NMS is a trademark of Novell, Inc.

Novell is a registered trademark of Novell, Inc., in the United States and other countries.

Novell Application Launcher is a trademark of Novell, Inc.

Novell Authorized Service Center is a service mark of Novell, Inc.

Novell Certificate Server is a trademark of Novell, Inc.

Novell Client is a trademark of Novell, Inc.

Novell Cluster Services is a trademark of Novell, Inc.

Novell Directory Services is a registered trademark of Novell, Inc.

Novell Distributed Print Services is a trademark of Novell, Inc.

Novell iFolder is a registered trademark of Novell, Inc.

Novell Labs is a trademark of Novell, Inc.

Novell SecretStore is a registered trademark of Novell, Inc.

Novell Security Attributes is a trademark of Novell, Inc.

Novell Storage Services is a trademark of Novell, Inc.

Novell, Yes, Tested & Approved logo is a trademark of Novell, Inc.

Nsure is a registered trademark of Novell, Inc.

Nterprise is a trademark of Novell, Inc.

Nterprise Branch Office is a trademark of Novell, Inc.

ODI is a trademark of Novell, Inc.

Open Data-Link Interface is a trademark of Novell, Inc.

Packet Burst is a trademark of Novell, Inc.

PartnerNet is a registered service mark of Novell, Inc., in the United States and other countries.

Printer Agent is a trademark of Novell, Inc.

QuickFinder is a trademark of Novell, Inc.

Red Box is a trademark of Novell, Inc.

Red Carpet is a registered trademark of Novell, Inc., in the United States and other countries.

Sequenced Packet Exchange is a trademark of Novell, Inc.

SFT and SFT III are trademarks of Novell, Inc.

SPX is a trademark of Novell, Inc.

Storage Management Services is a trademark of Novell, Inc.

SUSE is a registered trademark of SUSE AG, a Novell business.

System V is a trademark of Novell, Inc.

Topology Specific Module is a trademark of Novell, Inc.

Transaction Tracking System is a trademark of Novell, Inc.

TSM is a trademark of Novell, Inc.

TTS is a trademark of Novell, Inc.

Universal Component System is a registered trademark of Novell, Inc.

Virtual Loadable Module is a trademark of Novell, Inc.

VLM is a trademark of Novell, Inc.

Yes Certified is a trademark of Novell, Inc.

ZENworks is a registered trademark of Novell, Inc., in the United States and other countries.

Third-Party Materials

All third-party trademarks are the property of their respective owners.

Contents

About This Guide 9

1 Internationalization Concepts 11

1.1 Locale . . . 11
1.1.1 Native Locale . . . 11
1.1.2 Locale-Sensitive Data . . . 11
1.1.3 NetWare Locale Information Additions . . . 12
1.2 Function Groupings . . . 13
1.2.1 ANSI-based Internationalization Functions . . . 13
1.2.2 Internationalized Collation Functions . . . 13
1.2.3 Internationalized ctype.h Functions . . . 14
1.2.4 Internationalized Money Functions . . . 14
1.2.5 Internationalized printf Functions . . . 14
1.2.6 Internationalized String Functions . . . 14
1.2.7 Locale Information Functions . . . 15
1.2.8 Multibyte and Double-Byte Functions . . . 16
1.2.9 NLM Specific Functions . . . 16
1.2.10 Parameter Reordering Functions . . . 16

2 Internationalization Tasks 19

2.1 Setting up Internationalization Include Files . . . 19
2.2 Setting up for International String Manipulation . . . 19

3 Internationalization Functions 21

AddLanguage . . . 23
GetCurrentOSLanguageID . . . 25
LoadLanguageMessageTable . . . 26
localeconv . . . 28
NWatoi . . . 29
NWCharLwr . . . 31
NWCharType . . . 32
NWCharUpr . . . 33
NWCharVal . . . 34
NWGetCollateTable . . . 35
NWGetNWLOCALEVersion . . . 37
NWIncrement . . . 38
NWisalnum . . . 40
NWisalpha . . . 42
NWisdigit . . . 43
NWisxdigit . . . 45
NWitoa . . . 46
NWLInsertChar . . . 48
NWLlocaleconv . . . 49
NWLmblen . . . 51
NWLmbslen . . . 53
NWLsetlocale . . . 54

NWLstrbcpy . . . 56
NWLstrchr . . . 58
NWLstrcoll . . . 60
NWLstrcspn . . . 62
NWLstrftime . . . 64
NWLstricmp . . . 66
NWLstrlwr . . . 68
NWLstrpbrk . . . 69
NWLstrrchr . . . 71
NWLstrrev . . . 73
NWLstrspn . . . 75
NWLstrstr . . . 77
NWLstrtok . . . 79
NWLstrtok_r . . . 81
NWLstrupr . . . 83
NWLstrxfrm . . . 84
NWltoa . . . 86
NWLTruncateString . . . 88
NWNextChar . . . 89
NWPrevChar . . . 91
NWprintf . . . 93
NWsprintf . . . 95
NWstrImoney . . . 97
NWstrlen . . . 99
NWstrmoney . . . 100
NWstrncoll . . . 102
NWstrncpy . . . 104
NWstrnum . . . 106
NWultoa . . . 108
NWutoa . . . 110
NWvprintf . . . 112
NWvsprintf . . . 114
NWwsprintf (obsolete-moved from .h file 11/99) . . . 116
RenameLanguage . . . 117
ReturnLanguageName . . . 118
SetCurrentOSLanguageID . . . 119
setlocale . . . 120

4 Internationalization Structures 123

lconv . . . 124
LCONV . . . 127

A Revision History 131

About This Guide

Internationalization provides the functionality for adapting programs to various languages and localities and allows you to manipulate strings to use the internationally recognized Unicode standard. Functions are provided to support both single and multi-byte characters, as well as to support various locale conventions, such as decimal separations and time format.

Unicode is closely associated with internationalization as well.

This guide contains the following sections:

- Chapter 1, “Internationalization Concepts,” on page 11
- Chapter 2, “Internationalization Tasks,” on page 19
- Chapter 3, “Internationalization Functions,” on page 21
- Chapter 4, “Internationalization Structures,” on page 123

Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the User Comments feature at the bottom of each page of the online documentation.

Documentation Updates

For the most recent version of this guide, see NLM and NetWare Libraries for C (including CLIB and XPlat) (http://developer.novell.com/ndk/clib.htm)

Additional Information

For information about other CLib and XPlat interfaces, see the following guides:

- NLM Development Concepts, Tools, and Functions (http://developer.novell.com/ndk/doc/clib/ndev_enu/data/hqgkbkjr.html)
- Program Management (http://developer.novell.com/ndk/doc/clib/prog_enu/data/h9qu926c.html)
- NLM Threads Management (http://developer.novell.com/ndk/doc/clib/thmp_enu/data/h7g6q8vc.html)
- Connection, Message, and NCP Extensions (http://developer.novell.com/ndk/doc/clib/cmgnxenu/data/hvxfva0i.html)
- Multiple and Inter-File Services (http://developer.novell.com/ndk/doc/clib/mlti_enu/data/hby40vgi.html)
- Single and Intra-File Services (http://developer.novell.com/ndk/doc/clib/sngl_enu/data/h68b1qom.html)
- Volume Management (http://developer.novell.com/ndk/doc/clib/vol__enu/data/h6ixme3w.html)
- Client Management (http://developer.novell.com/ndk/doc/clib/clnt_enu/data/he77rked.html)

- Network Management (http://developer.novell.com/ndk/doc/clib/nwrk_enu/data/hvyko5s2.html)
- Server Management (http://developer.novell.com/ndk/doc/clib/srvr_enu/data/hzvjztxz.html)
- Unicode (http://developer.novell.com/ndk/doc/clib/ucod_enu/data/hjg275fp.html)
- Sample Code (http://developer.novell.com/ndk/doc/clib/code_enu/data/hwtnc7wc.html)
- Getting Started with NetWare Cross-Platform Libraries for C (http://developer.novell.com/ndk/doc/clib/startenu/data/hv9aw5v8.html)
- Bindery Management (http://developer.novell.com/ndk/doc/clib/bind_enu/data/h9qzn7u5.html)

For CLib source code projects, visit Forge (http://forge.novell.com).

For help with CLib and XPlat problems or questions, visit the NLM and NetWare Libraries for C (including CLIB and XPlat) Developer Support Forums (http://developer.novell.com/ndk/devforums.htm). There are two for NLM development (XPlat and CLib) and one for Windows XPlat development.

Documentation Conventions

In this documentation, a greater-than symbol (>) is used to separate actions within a step and items within a cross-reference path.

A trademark symbol (®, ™, etc.) denotes a Novell trademark. An asterisk (*) denotes a third-party trademark.

Internationalization Concepts

1

This documentation describes Internationalization, its functions, and features.

Internationalization is the strategy for adapting programs to diverse character sets and localities. The standard C contribution to internationalization is the header file locale.h. This file defines the lconv structure that controls formatting of numeric values for things like time and monetary expressions.

You can use internationalization functions declared in nwlocale.h in place of the standard C services found in locale.h. By using nwlocale.h, you not only receive the adaptive numeric formatting associated with locale.h, but also gain support for double-byte encoding schemes used to support large Japanese character sets. Because of the pervasive effect of a locale on standard input and output, internationalization functions in nwlocale.h also replace the print and scan functions declared in stdio.h.

1.1 Locale

A *locale* is the area of the world in which an application user's native language is spoken. Most of the time, a locale is a certain country: United States, Denmark, South Korea, etc.

1.1.1 Native Locale

The native local is the format in which the system presents certain common pieces of information—date, time, numbers, currency—which may vary from locale to locale.

In a NetWare server environment, the current locale is determined at OS boot-time. The locale cannot be altered without rebooting the OS.

To check which native locale your server is set for, use NWLsetlocale.

1.1.2 Locale-Sensitive Data

Use Internationalization to enable categories of information within a locale. Conceivably, an application may need to internationalize some categories of information and not others. Each category can be set individually. When you specify a category of locale information, only the functions related to that category are affected. For example, if you set the locale for LC_CTYPE, only the functions defined by ctype.h are internationalized. The following categories are defined:

Table 1-1 *Locale Categories*

LC_ALL (client only) NLC_ALL (client and NLM)	Set all categories
LC_COLLATE (client only) NLC_COLLATE (client and NLM)	Set only data related to collation functions
LC_CTYPE (client only) NLC_CTYPE (client and NLM)	Set only data related to ctype.h

LC_MONETARY (client only) NLC_MONETARY (client and NLM)	Set only data related to currency
LC_NUMERIC (client only) NLC_NUMERIC (client and NLM)	Set only data related to numeric format
LC_TIME (client only) NLC_TIME (client and NLM)	Set only data related to time format

The following table shows the various sequences for formatting information and the separators used to mark numeric divisions in three locales. Note that locales can share the same formatting conventions in some categories and not others.

Table 1-2 *Formatting Conventions for Three Locales*

Locale	Date	Time	Number	Currency
Australia	23/8/94	15:12:28	1,234.22	$1.22
Denmark	23-08-94	15.12.28	1.234,22	1,22 kr
South Korea	94.8.23	3:12:28 PM	1,234.22	W1.22

In Australia, the commonly used formats resemble those of the United States except for the time format. The time format is 24-hour and therefore does not need "AM" or "PM" in order to be understood. Notice that Denmark's time is also 24-hour, but South Korea's is 12-hour.

Notice that the date formats, including dividers, for the three countries are all different: Australia's is day/month/year, Denmark's is day-month-year, and South Korea's is year.month.day.

Number formats are important because the position of the comma and period frequently switch from locale to locale.

1.1.3 NetWare Locale Information Additions

In addition to the fields found in the lconv structure, the LCONV structure adds the following information.

Field	Type	Comment
code_page	int	Code page ID value.
country_id	int	Country code ID value.
data_list_separator	char[2]	Character used to separate items in a list.
date_separator	char[2]	Character used to separate date values.
time_separator	char[2]	Character used to separate time values.
time_format	char	0 = 12 hour, 1 = 24 hour clock
date_format	int	0 = MDY, 1 = DMY, 2 = YMD
reserved	char[40]	Undefined.

1.2 Function Groupings

text goes here

1.2.1 ANSI-based Internationalization Functions

The following table lists all the ANSI functions that are used for internationalizing NetWare applications. A few of these have been adapted to special Novell® conditions, but most of them have remained the same.

NetWare Function	ANSI-based Function
NWLlocaleconv	localeconv
NWLmblen	mblen
NWLsetlocale	setlocale
NWLstrchr	strchr
NWLstrcoll	strcoll
NWLstrcspn	strcspn
NWLstrftime	strftime
NWLstrpbrk	strpbrk
NWLstrrchr	strrchr
NWLstrrev	strrev
NWLstrspn	strspn
NWLstrstr	strstr
NWLstrupr	strupr
NWLstrxfrm	strxfrm

An example of an ANSI function that has been adapted for NetWare is NWLlocaleconv. The amount of information it provides has been increased. Also, the structure element data types have been changed from pointers to character arrays; in enhanced mode, Windows cannot tolerate them as pointers.

For ANSI descriptions of these ANSI-based internationalization functions, refer to the specification *ANSI X3.159-1989*.

1.2.2 Internationalized Collation Functions

These functions perform collation operations and are affected by the LC_COLLATE locale data. Use them in place of their counterparts in string.h.

NWGetCollateTable	Gets the Character Collation Table from the current operating system.
NWLstrcoll	Does a locale-sensitive string comparison of two strings.

NWstrncoll	Returns the difference in weight value between a string for which the collation value is known and a string for which it's not known.
NWLstrxfrm	Transforms a string by replacing each character with its corresponding collation value.

1.2.3 Internationalized ctype.h Functions

These functions perform alphanumeric tests and conversions. Use them in place of their counterparts in ctype.h.

NWatoi	Converts a string of digits into an integer value. (Double-byte characters aren't converted.)
NWisalnum	Returns whether a given value is alphabetic (A to Z or a to z) or a digit (0 to 9).
NWisalpha	Returns whether a given value is alphabetic (A to Z or a to z).
NWisdigit	Returns whether a given value is a digit (0 to 9).
NWIsxdigit	Returns whether a given value is a hexadecimal digit.

1.2.4 Internationalized Money Functions

These functions return monetary formats. Monetary formats are determined by the LC_MONETARY category of locale data.

NWstrImoney	Gets the country prefix and money format for a numerical value.
NWstrmoney	Gets the locale-sensitive money format for a numerical value.

1.2.5 Internationalized printf Functions

These functions perform printf -style operations. They provide for the reordering of parameters on the stack before outputting data. Use them in place of their counterparts in stdio.h.

NetWare Function	ANSI printf Function
NWprintf	printf
NWfprintf	fprintf
NWsprintf	sprintf
NWvprintf	vprintf
NWvfprintf	vfprintf
NWvsprintf	vsprintf

1.2.6 Internationalized String Functions

These functions perform string operations. Use them in place of their counterparts in string.h.

NWLstrbcpy	Copies a locale-sensitive string for a specified number of bytes (not characters).
NWLstrchr	Finds a character in a string.
NWLstrcspn	Computes the length of the maximum initial segment of one string consisting entirely of the characters not from another string.
NWLstrftime	Formats time and date according to the specified format.
NWLstricmp (client)	Performs a case-sensitive comparison of two strings.
NWstrlen (client)	Returns the number of bytes in a string.
NWLstrlwr (client)	Converts a string to lower case using locale information.
NWstrncpy	Copies a locale-sensitive string for a specified number of characters (not bytes).
NWstrnum	Formats a number for a specific country and returns the number in a string.
NWLstrpbrk	Locates the first occurrence in a string of any character from another string.
NWLstrrchr	Locates the last occurrence of a character in a string.
NWLstrrev	Reverses the order of the characters in a string.
NWLstrspn	Computes the length of the maximum initial segment of one string consisting entirely of characters from another string.
NWLstrstr	Searches one string for another string.
NWLstrtok (client)	Finds the next token in a string.
NWLstrupr	Converts a string to upper case using locale information.
NWLTruncateString (client)	Truncates a string at the specified number of characters.
NWultoa	Converts a long unsigned integer to a string.
NWutoa	Converts an unsigned integer to a string.
NWitoa	Converts an integer to a string.
NWltoa	Converts a long integer to a string.
NWLmbslen (client)	Counts the number of characters (not bytes) in a string.

1.2.7 Locale Information Functions

These functions access LCONV to initialize the locale.

NWGetNWLOCALEVersion (client only)	Returns the library version.
NWLlocaleconv	Sets the elements of an LCONV structure according to the current locale setting.
NWLsetlocale	Initializes the locale information.

NWLmblen	Counts the characters (not bytes) in a string.

1.2.8 Multibyte and Double-Byte Functions

These functions scan characters in multibyte strings.

NWCharType	Determines whether a character is single- or double-byte.
NWCharLwr (client only)	Converts a character to lower case. The result is based on the country information table.
NWCharUpr	Converts a character to upper case. The result is based on the country information table.
NWCharVal	Gets the integer value of a character in a string. Handles double-byte characters correctly.
NWIncrement	Increments a multibyte string pointer by the specified number of characters.
NWLInsertChar (client only)	Inserts a character at a given position in a string.
NWNextChar	Increments a pointer to the next character in a multibyte string.
NWPrevChar	Finds the beginning of the nearest previous character in a multibyte string.

See dbparse.c (../../../samplecode/clib_sample/intl_dbparse/dbparse.c.html) for sample code.

1.2.9 NLM Specific Functions

The functions in the following table are specific to NLM applications.

AddLanguage	Adds a language to the OS supported language list See Adding to OS Supported Language List: Example (*NDK: Sample Code*).
GetCurrentOSLanguageID	Returns the language ID for the language currently running on the server
LoadLanguageMessageTable	Loads a pointer to a language message table so that the table can be used for language enabling support
RenameLanguage	Changes the name string associated with an OS language ID
ReturnLanguageName	Returns the name associated with an OS language ID
SetCurrentOSLanguageID	Sets the language ID for the language currently running on the server

1.2.10 Parameter Reordering Functions

The following NetWare internationalization functions, defined in nwlocale.h, support parameter reordering:

- NWprintf (page 93)

- NWsprintf (page 95)
- NWvsprintf (page 114)

In addition, the following functions, defined in nwsnut.h support parameter reordering in for the *NLM User Interface Developer Components* (http://developer.novell.com/ndk/unsupported.htm#nwsnut):

- NWSAlert
- NWSAlertWithHelp
- NWSDisplayErrorText
- NWSDisplayErrorCondition
- NWSTrace

Internationalization Tasks

2

This documentation describes common tasks associated with Internationalization.

2.1 Setting up Internationalization Include Files

Include nwlocal.h in each application that calls a Novell internationalization function.

If your NetWare 3.x NLM uses register-based parameter-passing, include cdecl.h, to which the Novell internationalization functions have been added.

NOTE: For NetWare 4.x, 5.x, and 6.x, the "NWL" functions are provided by clib.nlm. For NetWare 3.12 these functions are provided by after311.nlm. The "NWL" functions also work on NetWare 3.11 if after311.nlm is loaded, but the end user might not have after311.nlm, since after311.nlm does not ship with the 3.11 OS.

2.2 Setting up for International String Manipulation

Setting up for international string manipulation involve the following steps:

1. Call NWLsetlocale.

   This function initializes internal data structures according to the currently implementation-defined native locale. It returns the country code of the native locale for which the server is currently set.

   While C programs use the "C" locale by default, an application can designate the machine's native locale with NWLsetlocale. A number of standard functions are sensitive to the current locale setting. For example, when the locale is changed, the functions in ctype.h change how they check characters.

   IMPORTANT: Because NWLsetlocale initializes the structures containing the locale information, *do not* call any other Internationalization function until you have called NWLsetlocale.

   ---

2. Call NWLlocaleconv

   If the Internationalization functions themselves do not provide all the locale-sensitive operations needed, you can call NWLlocaleconv to get raw locale data to write your own locale-sensitive functions.

   In addition, if you do not like the format for the locale as it is currently arranged, you can change it with NWLlocaleconv.

3. Replace existing functions with Internationalization functions

   See "ANSI Compatibility Functions" for the NetWare functions you should replace the ANSI-based functions with.

Internationalization Functions

3

This documentation alphabetically lists the Internationalization functions and describes their purpose, syntax, parameters, and return values.

- “AddLanguage” on page 23
- “GetCurrentOSLanguageID” on page 25
- “LoadLanguageMessageTable” on page 26
- “localeconv” on page 28
- “NWatoi” on page 29
- “NWCharLwr” on page 31
- “NWCharType” on page 32
- “NWCharUpr” on page 33
- “NWCharVal” on page 34
- “NWGetCollateTable” on page 35
- “NWGetNWLOCALEVersion” on page 37
- “NWIncrement” on page 38
- “NWisalnum” on page 40
- “NWisalpha” on page 42
- “NWisdigit” on page 43
- “NWisxdigit” on page 45
- “NWitoa” on page 46
- “NWLInsertChar” on page 48
- “NWLlocaleconv” on page 49
- “NWLmblen” on page 51
- “NWLmbslen” on page 53
- “NWLsetlocale” on page 54
- “NWLstrbcpy” on page 56
- “NWLstrchr” on page 58
- “NWLstrcoll” on page 60
- “NWLstrcspn” on page 62
- “NWLstrftime” on page 64
- “NWLstricmp” on page 66
- “NWLstrlwr” on page 68
- “NWLstrpbrk” on page 69
- “NWLstrrchr” on page 71
- “NWLstrrev” on page 73
- “NWLstrspn” on page 75

- “NWLstrstr” on page 77
- “NWLstrtok” on page 79
- “NWLstrtok_r” on page 81
- “NWLstrupr” on page 83
- “NWLstrxfrm” on page 84
- “NWltoa” on page 86
- “NWLTruncateString” on page 88
- “NWNextChar” on page 89
- “NWPrevChar” on page 91
- “NWprintf” on page 93
- “NWsprintf” on page 95
- “NWstrImoney” on page 97
- “NWstrlen” on page 99
- “NWstrmoney” on page 100
- “NWstrncoll” on page 102
- “NWstrncpy” on page 104
- “NWstrnum” on page 106
- “NWultoa” on page 108
- “NWutoa” on page 110
- “NWvprintf” on page 112
- “NWvsprintf” on page 114
- “NWwsprintf (obsolete-moved from .h file 11/99)” on page 116
- “RenameLanguage” on page 117
- “ReturnLanguageName” on page 118
- “SetCurrentOSLanguageID” on page 119
- “setlocale” on page 120

AddLanguage

Adds a language to the OS supported language list

Local Servers: nonblocking

Remote Servers: N/A

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM

Service: Internationalization

Syntax

```
#include <nwadv.h>

int AddLanguage  (
   int    languageID,
   BYTE  *newLanguageName,
   int    showErrorsToConsole);
```

Parameters

languageID

(IN) Specifies the language ID number for the language to be added (0-99 is reserved by the NetWare® OS.

newLanguageName

(IN) Points to the name of the language to be added.

showErrorsToConsole

(IN) Specifies whether to show error messages to the console screen.

Return Values

0	Success
1	Invalid ID or name

Remarks

AddLanguage adds the new language name and ID number to the OS language list.

See Adding to OS Supported Language List: Example (*NDK: Sample Code*).

See Also

GetCurrentOSLanguageID (page 25), LoadLanguageMessageTable (page 26), ReturnLanguageName (page 118), RenameLanguage (page 117), SetCurrentOSLanguageID (page 119)

GetCurrentOSLanguageID

Returns the language ID for the language currently running on the server

Local Servers: nonblocking

Remote Servers: N/A

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM

Service: Internationalization

Syntax

```
#include <nwadv.h>

int GetCurrentOSLanguageID  (
   void);
```

Return Values

Returns the language ID of the language for the OS.

See Also

AddLanguage (page 23), LoadLanguageMessageTable (page 26), RenameLanguage (page 117), ReturnLanguageName (page 118), SetCurrentOSLanguageID (page 119)

LoadLanguageMessageTable

Loads a pointer to a language message table so the table can be used for language enabling support

Local Servers: nonblocking

Remote Servers: N/A

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM

Service: Internationalization

Syntax

```
#include <nwadv.h>

LONG LoadLanguageMessageTable  (
   char***messageTable,
   LONG  *messageCount,
   LONG  *languageID);
```

Parameters

`messageTable`

(IN/OUT) Points to an array of pointers containing the Language Module file created with the MsgTools utility.

`messageCount`

(OUT) Points to the number of messages in the `messageTable` parameter.

`languageID`

(OUT) Points to the language ID number of the language of the `messageTable` parameter.

Return Values

0	Successful
-1	Unsuccessful

Remarks

NLM™ applications can be enabled for multiple languages by calling LoadLanguageMessageTable. Default language tables can be bound to NLM applications with the MESSAGES option in NLMLINK or NLMLINKX.

See Also

AddLanguage (page 23), GetCurrentOSLanguageID (page 25), RenameLanguage (page 117), ReturnLanguageName (page 118), SetCurrentOSLanguageID (page 119)

localeconv

Sets the components of an object with values appropriate for the formatting of numeric quantities according to the current locale

Local Servers: nonblocking

Remote Servers: N/A

NetWare Server: ANSI

Platform: NLM

Service: Internationalization

Syntax

```
#include <locale.h>

struct lconv *localeconv  (void);
```

Return Values

Returns a pointer to the filled-in object.

Remarks

localeconv sets the components of a struct lconv object with values appropriate for the formatting of numeric quantities according to the current locale. See the lconv (page 124) structure.

See Also

NWLlocaleconv (page 49), NWLsetlocale (page 54), setlocale (page 120)

NWatoi

Converts a string to an integer

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

nint N_API NWatoi (
   const nstr N_FAR  *string);
```

Pascal Syntax

```
uses netwin32

Function NWatoi
  (string : pnstr
) : nint;
```

Parameters

`string`

(IN) Points to the string of digits to be converted into an integer value.

Return Values

0x0000	No digit was converted
non-zero	Value converted from string

Remarks

NWatoi returns 0 if any double-byte characters are in the string.

NWatoi coverts a string of digits into an integer value. The syntax of the string must be:

```
long ::= [isspace]*[sign]digit[digit]*
```

IMPORTANT: Only decimal integers will work with NWatoi.

Tabs and/or spaces can precede the digits to be converted. A plus (+) or minus (-) sign can immediately precede the digits to be converted. Otherwise, any non-digit character terminates the conversion. Currently no double-byte characters are converted.

If the string cannot be converted to an integer, 0x0000 will be returned.

See Also

NWisdigit (page 43)

NWCharLwr

Converts a character to lower case

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <nwlocale.h>

nint N_API NWCharLwr  (
   nint   chr);
```

Pascal Syntax

```
uses netwin32

Function NWCharLwr
  (ch : nint
) : nint;
```

Parameters

chr

(IN) Specifies the character to be converted to lower case.

Return Values

Returns the lower case value of `chr`.

Remarks

Only single-byte characters are returned.

See Also

NWCharUpr (page 33), NWLsetlocale (page 54), NWLstrupr (page 83)

NWCharType

Determines whether a character is single- or double-byte

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

nint N_API NWCharType  (
   nint   ch);
```

Pascal Syntax

```
uses netwin32

Function NWCharType
  (ch : nint
) : nint;
```

Parameters

ch

(IN) Indicates the character to be tested (one or two bytes).

Return Values

0x0001	NWSINGLE_BYTE
0x0002	NWDOUBLE_BYTE

See Also

NWLsetlocale (page 54)

NWCharUpr

Converts a character to uppercase

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

nint NWAPI NWCharUpr  (
   nint   chr);
```

Pascal Syntax

```
uses netwin32

Function NWCharUpr
  (ch : nint
) : nint;
```

Parameters

chr

(IN) Specifies the character to be converted to upper case.

Return Values

Returns the uppercase value of the `chr` parameter.

Remarks

Only single-byte characters are returned.

See Also

NWCharLwr (page 31), NWLsetlocale (page 54), NWLstrupr (page 83)

NWCharVal

Returns the integer value of a character or a character in a string

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

nint N_API NWCharVal  (
   const nstr N_FAR  *string);
```

Pascal Syntax

```
uses netwin32

Function NWCharVal
  (str : pnstr
) : nint;
```

Parameters

`string`

(IN) Points to a character in a string.

Return Values

Returns the integer value of character pointed to by the `string` parameter.

Remarks

NWCharVal is useful when comparing the values of two characters. Double-byte characters are handled correctly.

See Also

NWLsetlocale (page 54)

NWGetCollateTable

Gets the Character Collation Table from the current OS

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

nint N_API NWGetCollateTable  (
   pnstr    retCollateTable,
   size_t   maxLen);
```

Pascal Syntax

```
uses netwin32

Function NWGetCollateTable
  (retCollateTable : pnstr;
  maxLen : size_t
) : nint;
```

Parameters

retCollateTable

(IN) Points to the buffer receiving the collate table.

maxLen

(IN) Specifies the maximum buffer size.

Return Values

0x0000	Successful
non-zero	Operating System Error

Remarks

A collation table is a 256-byte array giving the collation weight for each character in the current code page. The collation weight indicates how characters should be sorted, which varies from country to country.

See Also

NWLsetlocale (page 54)

NWGetNWLOCALEVersion

Returns the library version

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

void N_API NWGetNWLOCALEVersion  (
   pnuint8   majorVersion,
   pnuint8   minorVersion,
   pnuint8   revisionLevel,
   pnuint8   betaReleaseLevel);
```

Pascal Syntax

```
uses netwin32

Function NWGetNWLOCALEVersion
  (majorVersion     : pnuint8;
   minorVersion     : pnuint8;
   revisionLevel    : pnuint8;
   betaReleaseLevel : pnuint8
);
```

Parameters

majorVersion

(OUT) Points to an ASCII string containing the NWLocale major version.

minorVersion

(OUT) Points to an ASCII string containing the NWLocale minor version.

revisionLevel

(OUT) Points to the revision level.

betaReleaseLevel

(OUT) Points to the beta level.

NWIncrement

Increments a multibyte string pointer by a specified number of characters

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

pnstr N_API NWIncrement (
   const nstr N_FAR  *string,
   size_t   numChars);
```

Pascal Syntax

```
uses netwin32

Function NWIncrement
  (str      : pnstr;
   numChars : size_t
) : pnstr;
```

Parameters

`string`

(IN) Points to the string to increment.

`numChars`

(IN) Specifies the number of characters to increment.

Return Values

NULL	Less than the `numChars` parameter
non-NULL	Pointer to the position of the `numChars` parameter past the starting point of the string

Remarks

NWIncrement increments by characters, not bytes.

See Also

NWLsetlocale (page 54), NWNextChar (page 89)

NWisalnum

Returns a nonzero value if the given character is a letter, a digit, or a double-byte character

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

nint N_API NWisalnum (
   nuint   ch);
```

Pascal Syntax

```
uses netwin32

Function NWisalnum
  (ch : nuint
) : nint;
```

Parameters

ch

(IN) Specifies the character to be checked to determine if it is an alphabetic or numeric character.

Return Values

0x0000	Not an alphabetic or numeric character
non-zero	An alphabetic or numeric character

Remarks

DBCS characters are treated as alphanumeric characters. All double-byte characters return a non-zero value.

See Also

NWisalpha (page 42), NWisdigit (page 43)

NWisalpha

Returns a nonzero value if the given character is a letter or DBCS

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

nint N_API NWisalpha (
   nuint   ch);
```

Pascal Syntax

```
uses netwin32

Function NWisalpha
  (ch : nuint
) : nint;
```

Parameters

ch

(IN) Specifies the character to be checked to determine if it is an alphabetic character.

Return Values

0x0000	Not an alphabetic character
non-zero	An alphabetic character

Remarks

All double-byte characters return a non-zero value.

See Also

NWisalnum (page 40), NWisdigit (page 43)

NWisdigit

Returns a nonzero value if the given character is a digit

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

nint N_API NWisdigit (
  nuint   ch);
```

Pascal Syntax

```
uses netwin32

Function NWisdigit
  (ch : nuint
) : nint;
```

Parameters

ch

(IN) Specifies the character to be checked to determine if it is a numeric character.

Return Values

0x0000	Not a numeric character
non-zero	A numeric character

Remarks

DBCS characters passed to NWisdigit should be byte-swapped to Intel lo-hi order.

Currently all double-byte characters return zero.

See Also

NWisalnum (page 40), NWisalpha (page 42), NWisxdigit (page 45)

NWisxdigit

Returns a nonzero value if the given character is a hexadecimal digit

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <nwlocale.h>

nint N_API NWisxdigit  (
  nuint   ch);
```

Pascal Syntax

```
uses netwin32

Function NWisxdigit
(ch : nint
) : nint;
```

Parameters

ch

(IN) Specifies the character to be checked to determine if it is a hexadecimal character.

Return Values

0x0000	Not a hexadecimal character
non-zero	A hexadecimal character

Remarks

DBCS characters passed to NWisxdigit should be byte-swapped to Intel lo-hi order.

Currently all double-byte characters return zero.

See Also

NWisalnum (page 40), NWisalpha (page 42), NWisdigit (page 43)

NWitoa

Converts an integer to a string

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <nwlocale.h>

pnstr N_API NWitoa  (
   nint   value,
   pnstr   string,
   nuint   radix);
```

Pascal Syntax

```
uses netwin32

Function NWitoa
  (value : nint;
   str : pnstr;
   radix : nuint
) : pnstr;
```

Parameters

value

(IN) Specifies the integer to be converted.

string

(OUT) Points to the string result.

radix

(IN) Specifies the base of the `value` parameter (range 2-36).

Return Values

Returns a pointer to the string.

Remarks

NWitoa converts the digits of the specified `value` parameter to a NULL-terminated character string and stores the results in the `string` parameter.

If the `radix` parameter equals 10 and the `value` parameter is negative, the first character of the returned string is the minus sign. Otherwise, the value is treated as an unsigned value.

NWitoa returns NULL if an invalid value is passed into the `radix` parameter.

The `string` parameter must be pointing to a buffer large enough to contain the number being converted.

See Also

NWltoa (page 86), NWultoa (page 108), NWutoa (page 110)

NWLInsertChar

Inserts a character at a given position in a string

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <nwlocale.h>

nint N_API NWLInsertChar (
   pnstr              src,
   const nstr N_FAR  *insertableChar);
```

Pascal Syntax

```
uses netwin32

Function NWLInsertChar
  (src : pnstr;
   insertableChar : pnstr
) : nint;
```

Parameters

src

(IN/OUT) Points to a NULL-terminated string where the character is to be inserted.

insertableChar

(IN) Points to the character to be inserted (single- or double-byte).

Return Values

Returns the size of the inserted character (1 or 2 bytes).

NWLlocaleconv

Sets the elements of the LCONV structure according to the current locale setting

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

LCONV N_FAR *N_API NWLlocaleconv  (
  LCONV  N_FAR   *lconvPtr);
```

Pascal Syntax

```
uses netwin32

Function NWLlocaleconv
  (lconvPtr : pLCONV
) : pLCONV;
```

Parameters

lconvPtr

(OUT) Points to the LCONV structure.

Return Values

Returns a pointer to the LCONV structure.

Remarks

Call NWLlocaleconv to get locale information to complete tasks not provided for by other Internationalization functions. NWLlocaleconv sets the elements in the specified structure according to the currently selected locale.

NWLlocaleconv is not an exact analog to the ANSI localeconv function because NWLlocaleconv contains information that is not ANSI-specific.

For a more detailed description of the ANSI elements in the above structure, refer to the ANSI specification, *ANSI X3.159-1989*.

See Also

localeconv (page 28), NWLsetlocale (page 54)

NWLmblen

Returns the length of a multibyte character

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <nwlocale.h>

nint N_API NWLmblen (
   const nstr N_FAR  *string,
   size_t             maxBytes);
```

Pascal Syntax

```
uses netwin32

Function NWLmblen
  (string : pnstr;
   count : size_t
) : nint;
```

Parameters

`string`

(IN) Points to the string containing a character to evaluate.

`count`

(IN) Specifies the maximum number of bytes to evaluate.

Return Values

If the `string` parameter is not NULL, returns the length (in bytes) of the multibyte character.

If the `string` parameter is NULL, returns zero.

If the string does not form a valid multibyte character within the first count characters, returns -1.

Remarks

Generally, `maxBytes` is set to 2 and the function returns 1 for a 1-byte or 2 for a 2-byte character.

Use the NWLmbslen function to count the number of character is a string.

See Also

NWLmbslen (page 53)

NWLmbslen

Returns the number of characters (not bytes) in a specified string

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <nwlocale.h>

N_EXTERN_LIBRARY (size_t) NWLmbslen (
   const nuint8  *string);
```

Pascal Syntax

```
uses netwin32

Function NWLmbslen
  (string : const
) : size_t;
```

Parameters

string

(IN) Points to a NULL-terminated string.

Return Values

Returns the number of characters in the string.

Remarks

The string may consist of both single- and double-byte characters.

See Also

NWLmblen (page 51), NWstrlen (page 99)

NWLsetlocale

Initializes the implementation-defined native locale

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

pnstr N_API NWLsetlocale  (
   nint               category,
   const nstr N_FAR  *locale);
```

Pascal Syntax

```
uses netwin32

Function NWLsetlocale
  (category     : nint;
   const locale : pnstr
) : pnstr;
```

Parameters

category

(IN) Specifies the locale categories.

locale

(IN) Points to a locale string.

Return Values

NULL	Failure
non-NULL	Pointer to the Country ID

Remarks

WARNING: Call NWLsetlocale before calling any locale-sensitive functions. Typically NWLsetlocale is called with the following parameters:

```
NWLsetlocale(LC_ALL,"");
```

After calling NWLsetlocale, all locale-sensitive functions will use the locale information set by NWLsetlocale.

`locale` can have the following values:

```
Initialize the implementation-defined native
         environment.
NULL     Query for the current locale country ID,
         without initializing the environment.
```

The `category` parameter and internal data structures can be specified and initialized for that category only using less initialization time. Functions affected by an uninitialized category are not called.

The `category` parameter can have the following values:

```
NLC_ALL          0
NLC_COLLATE      1
NLC_CTYPE        2
NLC_MONETARY     3
NLC_NUMERIC      4
NLC_TIME         5
NLC_TOTAL         6
```

The country ID is a three-digit string defined by IBM. These IDs are based on the international phone prefix for a given country. For example, USA is `001'. Finland is `358'. The NWCallsInit function will automatically call the functions, NWSetLocale and NWInitUnicodeTables.

See Also

NWInitUnicodeTables (Unicode), NWLlocaleconv (page 49)

NWLstrbcpy

Copies a locale-sensitive string into the target buffer and NULL-terminates the target string

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

N_EXTERN_LIBRARY(pnstr) NWLstrbcpy  (
   pnstr              dest,
   const nstr N_FAR  *src,
   size_t             maxlen);
```

Pascal Syntax

```
uses netwin32

Function NWLstrbcpy
  (dest : pnstr;
   const src : pnstr;
   maxlen: size_t
) : pnstr;
```

Parameters

dest

(OUT) Points to the string to copy to.

src

(IN) Points to the string to be copied.

maxlen

(IN) Specifies the maximum number of bytes that may be copied into the target buffer including the NULL terminator.

Return Values

NULL	The `dest` parameter and/or the `src` parameter is NULL

non-NULL	Pointer to the `dest` parameter

Remarks

NWLstrbcpy is similar to the ANSI standard strncpy function, except NWLstrbcpy will not pad the remaining space in the output buffer with zeroes as the strncpy function does.

NWLstrbcpy is similar to the NWstrncpy function, except NWLstrbcpy specifies the maximum bytes (not characters) to copy. NWLstrbcpy will also NULL-terminate the string.

If the source string is greater than or equal to the `maxlen` parameter in length, only the first n-1 bytes are copied to the target buffer followed by NULL.

NWLstrbcpy will not truncate a double-byte character. If the output buffer is not large enough to hold both bytes of a double-byte character, the string is terminated and neither byte will be copied.

See Also

NWLsetlocale (page 54), NWstrncpy (page 104)

NWLstrchr

Finds a character in a string

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

pnstr N_API NWLstrchr (
   const nstr N_FAR  *string,
   nint               find);
```

Pascal Syntax

```
uses netwin32

Function NWLstrchr  (
   str  : pnstr;
   find : nint
) : pnstr;
```

Parameters

string

(IN) Points to the string to search.

find

(IN) Specifies the character for which to search (single- or double-byte).

Return Values

NULL	Character not found
non-NULL	Pointer to the character found

Remarks

NWLstrchr is case-sensitive.

See Also

NWLsetlocale (page 54), NWLstrrchr (page 71), NWLstrstr (page 77)

NWLstrcoll

Compares two strings according to the rules of the current locale

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

nint N_API NWLstrcoll (
   const nstr N_FAR  *string1,
   const nstr N_FAR  *string2);
```

Pascal Syntax

```
uses netwin32

Function NWLstrcoll
  (string1 : pnstr;
   string2 : pnstr
) : nint;
```

Parameters

`string1`

(IN) Points to the first string to compare.

`string2`

(IN) Points to the second string to compare.

Return Values

When the `string1` / `string2` parameter combination is relative to current locale setting:

integer>0 `string1`**>** `string2`
integer=0 `string1` **=** `string2`
integer<0 `string1`**<** `string2`

Remarks

NWLstrcoll compares `string1' to `string2'. Both are interpreted as appropriate to the LC_COLLATE category of the current locale. For a locale-sensitive comparison of two strings, call NWLstrcoll instead of the strcmp function.

The comparison between `string1` and `string2` is case insensitive.

See Also

NWLsetlocale (page 54), NWLstricmp (page 66)

NWLstrcspn

Computes the segment length of a specified string containing characters not found in another string

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

N_EXTERN_LIBRARY(size_t) NWLstrcspn  (
   const nstr N_FAR  *string1,
   const nstr N_FAR  *string2);
```

Pascal Syntax

```
uses netwin32

Function NWLstrcspn
  (const string1 : pnstr;
   const string2 : pnstr
) : size_t;
```

Parameters

`string1`

(IN) Points to the string to examine for characters from the `string2` parameter.

`string2`

(IN) Points to the characters to look for in the `string1` parameter.

Return Values

0x0000	`string1` =0
non-zero	0-based byte position in the `string1` parameter of the first character which matches any character in the `string2` parameter

Remarks

NWLstrcspn computes the length (in bytes) of the maximum initial segment of the `string1` parameter consisting entirely of characters not in the `string2` parameter.

Either or both of the strings can have double-byte characters.

See Also

NWLstrspn (page 75), NWLstrpbrk (page 69)

NWLstrftime

Formats the time and date according to a specified format

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h>
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

N_EXTERN_LIBRARY(size_t) NWLstrftime  (
   pnstr                   dst,
   size_t                  max,
   const nstr N_FAR       *fmt,
   const struct tm N_FAR   *ptm);
```

Pascal Syntax

```
uses netwin32

Function NWLstrftime
   (dst : pnstr;
   max : size_t;
   const fmt : nstr;
   const ptm : struct [tm]
) : size_t;
```

Parameters

dst

(OUT) Points to the buffer to contain date and time (if the `string` parameter is NULL, the length of the time string is returned).

max

(IN) Specifies the maximum size of the buffer where the date and time string is placed.

fmt

(IN) Points to the specified format.

ptm

(IN) Points to the `tm` structure.

Return Values

0x0000	The `max` parameter is 0 or the `fmt` parameter is NULL
non-zero	Length of formatted time

Remarks

NWLstrftime places characters into the `dst` parameter as formatted by the `fmt` parameter.

The `fmt` parameter can have the following values:

Value	Effect
%c	Date and time representation appropriate for locale ("2/22/97 04:30:00 pm" for USA locale)
%x	Date representation for current locale ("02/22/97" for USA locale)
%X	Time representation for current locale; may be in 12-hour format according to locales format ("04:30:00 pm" for USA locale)

If the `dst` parameter is set to NULL, the length of the formatted string is still returned. The formatted string is not returned.

See Also

NWLsetlocale (page 54)

NWLstricmp

Performs a case-insensitive compare of two strings

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <nwlocale.h>

N_EXTERN_LIBRARY(nint) NWLstricmp  (
   const nstr N_FAR  *str1,
   const nstr N_FAR  *str2);
```

Pascal Syntax

```
uses netwin32

Function NWLstricmp
  (const str1 : pnstr;
   const str2 : pnstr
) : nint;
```

Parameters

str1

(IN) Points to the first string to compare.

str2

(IN) Points to the second string to compare.

Return Values

<0	If the `str1` parameter is less than the `str2` parameter.
=0	If the `str1` parameter is identical to the `str2` parameter.
>0	If the `str1` parameter is greater than the `str2` parameter.

Remarks

NWLstricmp compares uppercase versions of the `str1` and `str2` parameters and returns a value indicating their relationship.

NWLstricmp correctly handles double-byte characters.

NWLstricmp performs the comparison lexicographically using the code page value of the characters and does not use the locale sensitive collation sequence as the NWLstrcoll function. However, NWLstricmp is faster than the NWLstrcoll function and is efficient for comparing strings for equality.

See Also

NWLstrcoll (page 60), NWstrncoll (page 102)

NWLstrlwr

Converts a string to lower case using locale information

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <nwlocale.h>

pnstr N_API NWLstrlwr  (
   pnstr   string);
```

Pascal Syntax

```
uses netwin32

Function NWLstrlwr
  (str1 : pnstr
) : pnstr;
```

Parameters

`string`

(IN/OUT) Points to the string to be converted to lower case.

Return Values

Returns a pointer to the lower case `string` parameter.

Remarks

NWLstrlwr is sensitive to double-byte characters if the locale includes double-byte characters.

See Also

NWCharLwr (page 31), NWLsetlocale (page 54), NWLstrupr (page 83)

NWLstrpbrk

Locates the first occurrence in a specified string of given characters from another string

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

pnstr N_API NWLstrpbrk  (
   pnstr              string1,
   const nstr N_FAR  *string2);
```

Pascal Syntax

```
uses netwin32

Function NWLstrpbrk
  (string1       : pnstr;
   const string2 : pnstr
) : pnstr;
```

Parameters

`string1`

(IN) Points to the string to search.

`string2`

(IN) Points to the list of characters to search for in the `string1` parameter.

Return Values

NULL	No characters from the `string2` parameter are in the `string1` parameter
non-NULL	Pointer to the first matching character (indicates characters from the `string2` parameter are in the `string1` parameter)

Remarks

NWLstrpbrk is sensitive to double-byte characters if the locale includes double-byte characters.

See Also

NWLsetlocale (page 54), NWLstrcspn (page 62), NWLstrspn (page 75)

NWLstrrchr

Locates the last occurrence of a character in a string

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

pnstr N_API NWLstrrchr (
   const nstr N_FAR  *string,
   nint               find);
```

Pascal Syntax

```
uses netwin32

Function NWLstrrchr
  (str  : pnstr;
   find : nint
) : pnstr;
```

Parameters

`string`

(IN) Points to the string to search.

`find`

(IN) Specifies the character for which to search (single- or double-byte).

Return Values

NULL	The `string` parameter is NULL or no match is found
non-NULL	Pointer to the character specified by the `find` parameter (indicates a match is found)

Remarks

NWLstrrchr searches backwards for the character specified in the `find` parameter.

See Also

NWLsetlocale (page 54), NWLstrchr (page 58)

NWLstrrev

Reverses the order of the characters in a string

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

psntr N_API NWLstrrev  (
   pnstr   string1,
   pnstr   string2);
```

Pascal Syntax

```
uses netwin32

Function NWLstrrev
  (string1 : pnstr;
   string2 : pnstr
) : pnstr;
```

Parameters

string1

(OUT) Points to the reversed string.

string2

(IN) Points to the string to be reversed.

Return Values

Pointer to the reversed string.

Remarks

Double-byte characters are treated as single units.

See Also

NWLsetlocale (page 54)

NWLstrspn

Computes the segment length of a specified string consisting entirely of characters from another string

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

N_EXTERN_LIBRARY(size_t) NWLstrspn  (
   const nstr N_FAR  *string1,
   const nstr N_FAR  *string2);
```

Pascal Syntax

```
uses netwin32

Function NWLstrspn
  (const string1 : pnstr;
   const string2 : pnstr
) : size_t;
```

Parameters

`string1`

(IN) Points to the string to search for characters from the `string2` parameter.

`string2`

(IN) Points to the characters to look for in the `string1` parameter.

Return Values

0x0000	The `string1` or `string2` parameter is NULL
non-zero	0-based position of first non-matching character

Remarks

NWLstrspn computes the length (in bytes) of the maximum initial segment of the `string1` parameter consisting entirely of characters from the `string2` parameter.

Double-byte characters are treated as single units.

See Also

NWLsetlocale (page 54), NWLstrcspn (page 62), NWLstrpbrk (page 69)

NWLstrstr

Searches a specified string for a given character string

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

pnstr N_API NWLstrstr (
   const nstr N_FAR  *string,
   const nstr N_FAR  *searchString);
```

Pascal Syntax

```
uses netwin32

Function NWLstrstr
  (str : pnstr;
   searchString : pnstr
) : pnstr;
```

Parameters

`string`

(IN) Points to the string to be searched.

`searchString`

(IN) Points to the character string for which to search.

Return Values

NULL	If the string pointed to by the `searchString` parameter is not found within the `string` parameter, or the`searchString` parameter is NULL or an empty string
non-NULL	Points to the `string` parameter beginning with the characters specified in the `searchString` parameter

Remarks

Double-byte characters are handled properly.

Unlike the ANSI version, if the `searchString` parameter points to an empty string (has zero length), NWLstrstr will return NULL instead of a pointer to the `string` parameter.

See Also

NWLsetlocale (page 54), NWLstrchr (page 58)

NWLstrtok

Finds the next token in a specified string

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <nwlocale.h>

pnstr N_API NWLstrtok  (
   pnstr   parse,
   const nstr N_FAR  *delim);
```

Pascal Syntax

```
uses netwin32

Function NWLstrtok
  (parse : pnstr;
   delim : pnstr
) : pnstr;
```

Parameters

parse

(IN/OUT) Points to the string containing the token(s).

delim

(IN) Points to the string containing delimiters (can include double-byte characters).

Return Values

Returns a pointer to the next token found in the `parse` parameter.

Returns NULL when no more tokens are found.

Remarks

Every time NWLstrtok is called, it modifies the `parse` parameter by substituting a NULL character for each delimiter that is found.

Sequentially calling NWLstrtok breaks the string pointed to by the `parse` parameter into a sequence of tokens, each of which is delimited by a character from the string pointed to by the

`delim` parameter. The first time NWLstrtok is called in the sequence, the `parse` parameter is its first parameter. It will be followed by calling NWLstrtok additional times with a NULL pointer as the first parameter. The separator string pointed to by the `delim` parameter might be different each time NWLstrtok is called.

The first time in the sequence that NWLstrtok is called, it searches the string pointed to by the `parse` parameter for the first character that is NOT contained in the current separator string pointed to by the `delim` parameter. If no such character is found, there are no tokens in the string and NWLstrtok returns a NULL pointer. If such a character is found, the character is the start of the first token.

NWLstrtok continues searching, beginning at the token found for a character that IS contained in the current separator string. If no such character is found, the current token extends to the end of the string pointed to by the `parse` parameter. Subsequent searches for a token will return a NULL pointer. If such a character is found, it is overwritten by a NULL character, which terminates the current token. NWLstrtok saves a pointer to the following character, from which the next search for a token will start.

Each subsequent call, with a NULL pointer as the value of the first parameter, starts searching from the saved pointer as described.

WARNING: NWLstrtok uses a static variable for maintaining the last next token location. If multiple or simultaneous calls are made to NWLstrtok, a high potential for data corruption and incorrect results exists.

Do not attempt to call NWLstrtok simultaneously for different strings. Be aware of calling NWLstrtok from within a loop where another application might also be calling NWLstrtok.

In a multi-threaded, multi-processor environment, use the NWLstrtok_r function.

See Also

NWLstrspn (page 75), NWLstrcspn (page 62), NWLstrtok_r (page 81)

NWLstrtok_r

Finds the next token in a specified string

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <nwlocale.h>

pnstr N_API NWLstrtok_r  (
   pnstr              parse,
   const nstr N_FAR  *delim,
   ppnstr             last);
```

Parameters

parse

(IN/OUT) Points to the string containing the tokens.

delim

(IN) Points to the string containing delimiters, which can include double-byte characters.

last

(IN/OUT) Points to a value used to record the progress through the string specified by the `parse` parameter.

Return Values

If a delimiter is found, returns a pointer to the first byte of a token. On subsequent iterations, if no delimiter is found, returns a NULL pointer.

If `parse` does not contain any of the delimiters specified in `delim`, returns a pointer to `parse`, and all of `parse` is considered to be a token.

Remarks

Every time NWLstrtok_r is called, it modifies the `parse` parameter by substituting a NULL character for each delimiter that is found.

The first call to NWLstrtok_r returns a pointer to the first character of the first token in the string parameter and writes a NULL character into the string parameter immediately following the returned token. On subsequent calls, the `parse` parameter must be set to NULL, and NWLstrtok_r uses the value in the `last` parameter to search through the string and return successive tokens.

Because the NWLstrtok_r function can modify the string by writing a NULL character to terminate a token, the string should be duplicated if the string is to be reused.

See Also

NWLstrspn (page 75), NWLstrcspn (page 62), NWLstrtok (page 79)

NWLstrupr

Converts a string to uppercase using locale information

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

pnstr N_API NWLstrupr  (
   pnstr   string);
```

Pascal Syntax

```
uses netwin32

Function NWLstrupr
  (str : pnstr
) : pnstr;
```

Parameters

`string`

(IN/OUT) Points to the string to be converted to uppercase.

Return Values

Returns a pointer to the lower case `string` parameter.

Remarks

NWLstrupr is sensitive to double-byte characters if the locale includes double-byte characters.

See Also

NWCharUpr (page 33), NWLsetlocale (page 54), NWLstrlwr (page 68)

NWLstrxfrm

Transforms a string by replacing each character with its corresponding collation value

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

size_t N_API NWLstrxfrm (
   pnstr              string1,
   const nstr N_FAR  *string2,
   size_t             numBytes);
```

Pascal Syntax

```
uses netwin32

Function NWLstrxfrm
  (string1  : pnstr;
   string2  : pnstr;
   numBytes : size_t
) : size_t;
```

Parameters

string1

(OUT) Points to the string after it is transformed to its collation value (optional).

string2

(IN) Points to the string to be transformed.

numBytes

(IN) Specifies the size of the output buffer in bytes (including the NULL terminator).

Return Values

0x0000	String not transformed

non-zero	Length of the complete transformed string (not just the part of the string that fits in the output buffer).

Remarks

NWLstrxfrm transforms the `string2` parameter to the corresponding collation weights and places the results in the `string1` parameter.

Only the number of bytes specified by the `numBytes` parameter will be written to the `string1` parameter. If the `numBytes` parameter is zero, the `string1` parameter can be NULL.

To find the total number of bytes required to hold the transformed string and the NULL terminator, use the expression

```
(NWLstrxfrm(NULL, input, 0) +1)
```

If the return value plus one is greater than the number specified by the `numBytes` parameter, the transformed string did not fit in the output buffer and the output buffer might not be NULL terminated. If the return value plus one is less than or equal to the number specified by the `numBytes` parameter, the entire string (including the NULL terminator) is contained in the output buffer.

A performance advantage is provided by calling NWLstrxfrm in place of the NWLstrcoll function when multiple comparisons of the same string are necessary. If you call the strcmp function with two transformed strings, the result is similar to calling the NWLstrcoll function with the two original strings. By calling NWLstrxfrm, however, only the first byte of a double-byte character is used to determine the collation weight, while the NWLstrcoll function compares both bytes.

If the collation table is not loaded, the `numBytes` parameter is `0', or the `string2` parameter is NULL, zero will be returned.

NOTE: NWLstrxfrm does not distinguish between characters with accents and characters without accents unless the character is specifically in the alphabet for the country.

See Also

NWGetCollateTable (page 35)NWLsetlocale (page 54), NWLstrcoll (page 60)

NWltoa

Converts a long integer to a string

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <nwlocale.h>

pnstr N_API NWltoa  (
   nuint32   value,
   pnstr     buf,
   nuint     radix);
```

Pascal Syntax

```
uses netwin32

Function NWltoa
  (value : nint32;
   buf : pnstr;
   radix : nuint
) : pnstr;
```

Parameters

value

(IN) Specifies the long integer to be converted.

buf

(OUT) Points to a buffer to hold the string result.

radix

(IN) Specifies the base of the `value` parameter (range 2-36).

Return Values

Returns a pointer to the string.

Remarks

NWltoa converts the digits of the specified `value` parameter to a NULL-terminated character string and stores the results in the `buf` parameter.

If the `radix` parameter equals 10 and the `value` parameter is negative, the first character of the returned string is the minus sign. Otherwise, the value is treated as an unsigned value.

NWltoa returns NULL if an invalid value is passed into the `radix` parameter.

The `buf` parameter must be pointing to a buffer large enough to contain the number being converted.

See Also

NWitoa (page 46), NWultoa (page 108), NWutoa (page 110)

NWLTruncateString

Truncates a string at the specified number of bytes

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <nwlocale.h>

N_EXTERN_LIBRARY (nint) NWLTruncateString  (
   pnchar8   pStr,
   nint      iMaxLen);
```

Pascal Syntax

```
uses netwin32

Function NWltoa
  (pStr : pnchar8;
   iMaxLen : nint
) : nint;
```

Parameters

pStr

(IN) Points to the first character in the string (can contain double-byte characters).

iMaxLen

(IN) Specifies the maximum length of the string, including the NULL terminator, in bytes.

Return Values

Returns the length of the string (might be truncated) not counting the NULL terminator (in bytes).

Remarks

NWLTruncateString truncates the string if necessary so the entire string, including the NULL termination byte, will fit in a buffer of the number of bytes specified by the `iMaxLen` parameter.

If the truncation would chop a double-byte character in half, the first half of the double-byte character is also eliminated.

NWNextChar

Increments a pointer to the next character in a string with multibyte characters

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

pnstr N_API NWNextChar  (
   const nstr N_FAR  *string);
```

Pascal Syntax

```
uses netwin32

Function NWNextChar
  (const str : pnstr
) : pnstr;
```

Parameters

`string`

(IN) Points to the address of the current position in the string. If there is no next character, `string` points to the null character at the end of the string.

Return Values

Pointer to the next character (not byte) in the specified string.

Remarks

NWNextChar is called to move through strings whose characters are one or two bytes each in length. For example, NWNextChar could be called for strings containing characters from a Japanese character set.

In a multibyte string, it is not obvious whether the current byte is a single-byte character or the second character of a double-byte character. NWNextChar resolves this ambiguity.

Increment a pointer with he statement ptr=NWNextChar(ptr) instead of ptr++.

The NWNextChar function is implemented as a call to AnsiNext in Windows.

See Also

NWLsetlocale (page 54), NWIncrement (page 38), NWPrevChar (page 91)

NWPrevChar

Finds the beginning of the nearest previous character in a string with multibyte characters

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

pnstr N_API NWPrevChar  (
   const nstr N_FAR  *string,
   pnstr              position);
```

Pascal Syntax

```
uses netwin32

Function NWPrevChar
  (const str : pnstr;
   position : pnstr
) : pnstr;
```

Parameters

`string`

(IN) Points to the beginning of the string.

`position`

(IN) Points to a character in the NULL-terminated string.

Return Values

0x0000	The `string` parameter is NULL
non-zero	Pointer to the previous character in the string. If the `position` parameter is equal to the `string` parameter, this value points to the first character in the string

Remarks

In a multibyte string, it is not obvious whether the current byte is a single-byte character or the second character of a double-byte character. NWPrevChar resolves this ambiguity.

See Also

NWLsetlocale (page 54), NWNextChar (page 89)

NWprintf

Formats and outputs a string to stdout (parameter reordering is supported)

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

N_EXTERN_LIBRARY_C(nint) NWprintf  (
   const nstr N_FAR  *format,
                     [parameter_1,
                      parameter_2 ]...);
```

Pascal Syntax

```
uses netwin32

Function NWprintf
  (const format : nstr,
   parameter_1, parameter_2, ...
) : nint;
```

Parameters

format

(IN) Points to the format string determining how the data will be formatted before sending it to standard output.

parameter_1, parameter_2,. ..

(IN) Specifies the user-supplied variable list of parameters whose values are used in the formatted output.

Return Values

Returns the number of bytes output.

Remarks

The NWprintf function is the same as the standard C printf with the exception that it supports parameter reordering. This feature allows a language translator to change the order in which arguments are printed by just changing the format string.

There are two ways to do parameter reordering. The most common way is to specify the argument order with a %n in front of each formatting code. For example:

NWprintf("There are %d files in the directory %s.\n", numfiles, dirname);

In an internationalized program, the format string would actually be stored in a separate file to be translated into other languages. The following statement changes the order of the arguments with only a change in the format string:

NWprintf("Directory %2%s contains %1%d files.\n", numfiles, dirname);

The second method allows parameter ordering to be determined at run time. The format string is prepended with a reordering vector specifying the argument order. The reordering vector is a series of bytes with the following format:

LDH!<n><o1><o2><o3>...<format string> <n><o1><o2><o3> etc. are binary bytes.

Where:

<n> is the number of arguments in the printf statements. <o1> specifies which argument should be printed first. If the third argument should be printed first, then <o1> would contain the binary value 3. <o2> specifies which argument should be printed second, etc.

You can also use strings similar to the following:

```
printf ("Test: %3$s %2$d %1$s", string, 10, string);
```

See Also

NWsprintf (page 95), NWvsprintf (page 114), NWvprintf (page 112)

NWsprintf

Formats a string and outputs it to a user buffer (parameter reordering is supported)

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

N_EXTERN_LIBRARY_C(nint) NWsprintf  (
   pnstr              buffer,
   const nstr N_FAR  *format,
                     [parameter_1,
                      parameter_2 ]...);
```

Pascal Syntax

```
uses netwin32

Function NWsprintf
  (buffer : pnstr;
  const format : nstr,
   parameter_1, parameter_2, ...
) : nint;
```

Parameters

buffer

(OUT) Points to the buffer receiving the formatted data.

format

(IN) Points to the format string determining how the data will be formatted before sending it to the buffer.

parameter_1, parameter_2,. ..

(IN) Specifies the user-supplied variable list of parameters whose values are used in the formatted output.

Return Values

Returns the number of bytes output.

Remarks

For information about data formatting, see the printf function in any C manual.

If the NWsprintf function is the same as the C standard sprintf function except that parameter reordering is supported.

The reordering feature allows a language translator to change the order in which arguments are printed by just changing the format string.

There are two ways to do parameter reordering. The most common way is to specify the argument order with a %n in front of each formatting code. For example:

NWprintf("There are %d files in the directory %s.\n", numfiles, dirname);

In an internationalized program, the format string would actually be stored in a separate file to be translated into other languages. The following statement changes the order of the arguments with only a change in the format string:

NWprintf("Directory %2%s contains %1%d files.\n", numfiles, dirname);

The second method allows parameter ordering to be determined at run time. The format string is prepended with a reordering vector specifying the argument order. The reordering vector is a series of bytes with the following format:

LDH!<n><o1><o2><o3>...<format string> <n><o1><o2><o3> etc. are binary bytes.

Where:

<n> is the number of arguments in the printf statements. <o1> specifies which argument should be printed first. If the third argument should be printed first, then <o1> would contain the binary value 3. <o2> specifies which argument should be printed second, etc..

See Also

NWprintf (page 93), NWvsprintf (page 114), NWvprintf (page 112)

NWstrImoney

Returns the country prefix and money format for a numerical value

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

pnstr N_API NWstrImoney  (
   pnstr         buffer,
   NUMBER_TYPE   Value);
```

Pascal Syntax

```
uses netwin32

Function NWstrImoney
  (buffer : pnstr;
   Value : NUMBER_TYPE
) : pnstr;
```

Parameters

buffer

(OUT) Points to the number formatted for a specific country in the international format.

Value

(IN) Specifies the number to format.

Return Values

Returns a pointer to `string`.

Remarks

NWstrImoney has no ANSI counterpart.

The last 2 or 3 (some locales) digits of the `Value` parameter are always formatted as the smallest money units for that locale. The `Value` parameter is of the nint32 type.

NWstrImoney is different from the NWstrmoney function since it provides country prefixes in the `buffer` parameter.

The `buffer` parameter can have the following values:

```
BELGIUM        BEF
CANADA_FR      CAD
DENMARK        DKK
FINLAND        FIM
FRANCE         FRF
GERMANY        DDM
ITALY          ITL
NETHERLANDS    NLG
NORWAY         NOK
PORTUGAL       PTE
SPAIN          ESP
SWEDEN         SEK
SWITZERLAND    SFR
UK             GBP
USA            USD
JAPAN          JPY
KOREA          KRW
PRC            CNY
TAIWAN         TWD
```

Examples:

```
locale      number      formatted value

US          1234         USD 12.34
FRANCE      1234         FRF 12,34
```

For example, if country code 033 (France) were used and 3498 is passed in the `Value` function, the output string would be "FRF 34,98".

See Also

NWLsetlocale (page 54), NWstrmoney (page 100)

NWstrlen

Returns the number of bytes in a string

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <nwlocale.h>

N_EXTERN_LIBRARY(nint) NWstrlen  (
   const nstr N_FAR  *str);
```

Pascal Syntax

```
uses netwin32

Function NWstrlen
  (const str : pnstr
) : nint;
```

Parameters

str

(IN) Points to the NULL-terminated string to be counted.

Return Values

Returns the number of bytes (not characters) in a string (not including the NULL-termination byte).

See Also

NWLmbslen (page 53)

NWstrmoney

Returns the locale-sensitive money format for a numerical value

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

pnstr N_API NWstrmoney  (
   pnstr         buffer,
   NUMBER_TYPE   Value);
```

Pascal Syntax

```
uses netwin32

Function NWstrmoney
  (buffer : pnstr;
   Value : NUMBER_TYPE
) : pnstr;
```

Parameters

`buffer`

(OUT) Points to the number formatted for a specific country.

`Value`

(IN) Specifies the number to format.

Return Values

Returns a pointer to `string`.

Remarks

NWstrmoney has no ANSI counterpart. NWstrmoney is different from NWstrImoney ; it does not provide the country prefix in `string`.

The last 2 or 3 (some locales) digits of `Value` will always be formatted as the smallest money units for that locale. The type of this variable is long double.

Examples:

```
locale      number       formatted value
US           1234          $12.34
FRANCE       1234          p12,34
```

See Also

NWLsetlocale (page 54), NWstrImoney (page 97)

NWstrncoll

Returns the locale-sensitive comparison of two strings

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

nint N_API NWstrncoll (
   const nstr N_FAR  *string1,
   const nstr N_FAR  *string2,
   size_t             maxChars);
```

Pascal Syntax

```
uses netwin32

Function NWstrncoll
  (string1  : pnstr;
   string2  : pnstr;
   maxChars : size_t
) : nint;
```

Parameters

`string1`

(IN) Points to the first string to compare.

`string2`

(IN) Points to the second string to compare.

`maxChars`

(IN) Specifies the maximum number of characters to compare.

Return Values

0x0000	Two strings are identical or the `maxChars` parameter is zero

<0	The `string1` parameter is less than the `string2` parameter, relative to the current locale setting
>0	The `string1` parameter is greater than the `string2` parameter, relative to the current locale setting

Remarks

If no collate table exists, the locale is the C locale (see the strcmp function).

NWstrncoll is useful to insert or delete items in a sorted list based on their collation value.

Double byte characters count as one character in the count.

See Also

NWLsetlocale (page 54), NWLstrcoll (page 60), NWLstrxfrm (page 84)

NWstrncpy

Copies a locale-sensitive string for a specified number of characters (not bytes)

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

pnstr N_API NWstrncpy (
   pnstr              target_string,
   const nstr N_FAR  *source_string,
   nint               numChars);
```

Pascal Syntax

```
uses netwin32

Function NWstrncpy
  (target_string : pnstr;
   source_string : pnstr;
   numChars      : nint
) : pnstr;
```

Parameters

target_string

(OUT) Points to the string to which to copy.

source_string

(IN) Points to the string to be copied.

numChars

(IN) Specifies the number of characters to copy.

Return Values

NULL	The `target_string` parameter and/or the `source string` parameter is NULL
non-NULL	Pointer to the `target_string` parameter

Remarks

NWstrncpy copies a string of up to the number of characters specified by the `numChars` parameter to the destination string and NULL-terminates the string if the source string contains less characters than the number specified by the `numChars` parameter. However, unlike the ANSI strncpy function, NWstrncpy does not pad any remaining space in the destination string with NULLs.

To guarantee a NULL-terminated string, set the `numChars` parameter equal to the length of the `target_string` parameter minus one and set the last element of the `target_string` parameter equal to zero.

See Also

NWLsetlocale (page 54)

NWstrnum

Formats a number for a specific country and returns the number in a string

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

pnstr N_API NWstrnum  (
   pnstr         buffer,
   NUMBER_TYPE   Value);
```

Pascal Syntax

```
uses netwin32

Function NWstrnum
  (buffer : pnstr;
   Value  : NUMBER_TYPE
) : pnstr;
```

Parameters

`buffer`

(OUT) Points to the number formatted for a specific country.

`Value`

(IN) Specifies the number to format (long).

Return Values

Returns a pointer to the `string` parameter.

Remarks

If country code 033 (France) was used and 3498 was passed in the `Value` parameter, NWStrnum will return 3.498.

Examples:

```
locale      number      formatted value

US          1234            1,234
FRANCE      1234            1.234
```

See Also

NWLsetlocale (page 54)

NWultoa

Converts a long unsigned integer to a string

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <nwlocale.h>

pnstr N_API NWultoa  (
   nuint32   value,
   pnstr     buf,
   nuint     radix);
```

Pascal Syntax

```
uses netwin32

Function NWultoa
  (value : nuint32;
   buf : pnstr;
   radix : nuint
) : pnstr;
```

Parameters

value

(IN) Specifies the long unsigned integer to be converted.

buf

(OUT) Points to the string result.

radix

(IN) Specifies the base of the `value` parameter (range 2-36).

Return Values

Returns a pointer to the string.

Remarks

NWultoa converts the digits of the specified `value` parameter to a NULL-terminated character string and stores the results in the `buf` parameter.

NWultoa returns NULL if an invalid value is passed into the `radix` parameter.

The `buf` parameter must be pointing to a buffer large enough to contain the number being converted.

See Also

NWitoa (page 46), NWltoa (page 86), NWutoa (page 110)

NWutoa

Converts an unsigned integer to a string

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <nwlocale.h>

pnstr N_API NWutoa  (
   nuint   value,
   pnstr   string,
   nuint   radix);
```

Pascal Syntax

```
uses netwin32

Function NWultoa
  (value : nuint32;
   str : pnstr;
   radix : nuint
) : pnstr;
```

Parameters

value

(IN) Specifies the unsigned integer to be converted.

string

(OUT) Points to the string result.

radix

(IN) Specifies the base of the `value` parameter (range 2-36).

Return Values

Returns a pointer to the string.

Remarks

NWutoa converts the digits of the specified `value` parameter to a NULL-terminated character string and stores the results in the `string` parameter.

NWutoa returns NULL if an invalid value is passed into the `radix` parameter.

The `string` parameter must be pointing to a buffer large enough to contain the number being converted.

See Also

NWitoa (page 46), NWltoa (page 86), NWultoa (page 108)

NWvprintf

Formats a string and outputs it to standard output (parameter reordering is supported)

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

N_EXTERN_LIBRARY(nint) NWvprintf  (
   const nstr N_FAR  *format,
   va_list            argList);
```

Pascal Syntax

```
uses netwin32

Function NWvprintf
  (format : const nstr;
   argList : va_list
) : nint;
```

Parameters

format

(IN) Points to the format string determining how the data will be formatted before sending it to standard output.

argList

(IN) Specifies a variable list of parameters whose values are used in the formatted output.

Return Values

Returns the number of bytes output.

Remarks

The variable argument list is given as a single va_list parameter.

For information about formatting data, see the printf function in any C manual.

The reordering feature allows a language translator to change the order in which arguments are printed by just changing the format string.

There are two ways to do parameter reordering. The most common way is to specify the argument order with a %n in front of each formatting code. For example:

NWprintf("There are %d files in the directory %s.\n", numfiles, dirname);

In an internationalized program, the format string would actually be stored in a separate file to be translated into other languages. The following statement changes the order of the arguments with only a change in the format string:

NWprintf("Directory %2%s contains %1%d files.\n", numfiles, dirname);

The second method allows parameter ordering to be determined at run time. The format string is prepended with a reordering vector specifying the argument order. The reordering vector is a series of bytes with the following format:

LDH!<n><o1><o2><o3>...<format string> <n><o1><o2><o3> etc. are binary bytes.

Where:

<n> is the number of arguments in the printf statements. <o1> specifies which argument should be printed first. If the third argument should be printed first, then <o1> would contain the binary value 3. <o2> specifies which argument should be printed second, etc..

See Also

NWprintf (page 93), NWsprintf (page 95), NWvsprintf (page 114)

NWvsprintf

Formats a string and outputs it to standard output (parameter reordering is supported)

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM, Windows NT, Windows 95, Windows 98

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
#include <time.h> or #define NWL_EXCLUDE_TIME
#include <stdio.h> or #define NWL_EXCLUDE_FILE
#include <nwlocale.h>

N_EXTERN_LIBRARY(nint) NWvsprintf  (
   pnstr              buffer,
   const nstr N_FAR  *format,
   va_list            argList);
```

Pascal Syntax

```
uses netwin32

Function NWvsprintf
  (buffer : pnstr
  format : const nstr;
   argList : va_list
) : nint;
```

Parameters

buffer

(OUT) Points to the buffer receiving the formatted data.

format

(IN) Points to the format string determining how the data will be formatted before sending it to the buffer.

argList

(IN) Specifies a variable list of parameters whose values are used in the formatted output.

Return Values

Returns the number of bytes output.

Remarks

The variable argument list is given as a single va_list parameter.

For information about formatting data, see the printf function in any C manual.

The reordering feature allows a language translator to change the order in which arguments are printed by just changing the format string.

There are two ways to do parameter reordering. The most common way is to specify the argument order with a %n in front of each formatting code. For example:

NWprintf("There are %d files in the directory %s.\n", numfiles, dirname);

In an internationalized program, the format string would actually be stored in a separate file to be translated into other languages. The following statement changes the order of the arguments with only a change in the format string:

NWprintf("Directory %2%s contains %1%d files.\n", numfiles, dirname);

The second method allows parameter ordering to be determined at run time. The format string is prepended with a reordering vector specifying the argument order. The reordering vector is a series of bytes with the following format:

LDH!<n><o1><o2><o3>...<format string> <n><o1><o2><o3> etc. are binary bytes.

Where:

<n> is the number of arguments in the printf statements. <o1> specifies which argument should be printed first. If the third argument should be printed first, then <o1> would contain the binary value 3. <o2> specifies which argument should be printed second, etc..

See Also

NWprintf (page 93), NWsprintf (page 95), NWvprintf (page 112)

NWwsprintf (obsolete-moved from .h file 11/99)

Was last documented in September 1999. Call NWsprintf (page 95) instead.

RenameLanguage

Changes the name string associated with an OS language ID

Local Servers: nonblocking

Remote Servers: N/A

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM

Library: Cross-Platform Localization (LOC*.*)

Service: Internationalization

Syntax

```
incl <nwadv.h>

int RenameLanguage  (
   int    languageID,
   BYTE  *newLanguageName,
   int    showErrorsToConsole);
```

Parameters

languageID

(IN) Specifies the language ID number for the language to be renamed.

newLanguageName

(IN) Points to the new name to associate with the language ID.

showErrorsToConsole

(IN) Specifies whether to show error messages to the console screen.

Return Values

0	Success
1	Invalid ID or name

Remarks

RenameLanguage renames the language using the ID specified by the `languageID` parameter.

See Also

AddLanguage (page 23), GetCurrentOSLanguageID (page 25), LoadLanguageMessageTable (page 26), ReturnLanguageName (page 118), SetCurrentOSLanguageID (page 119)

ReturnLanguageName

Returns the name associated with an OS language ID

Local Servers: nonblocking

Remote Servers: N/A

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM

Service: Internationalization

Syntax

```
incl <nwadv.h>

int ReturnLanguageName  (
   int    languageID,
   BYTE  *languageName);
```

Parameters

`languageID`

(IN) Specifies the languageID for which to return the language name.

`languageName`

(OUT) Points to the language name associated with the specified language ID.

Return Values

0	Successful
1	Invalid language ID

Remarks

The `languageName` parameter receives the name associated with the `languageID` parameter.

See Also

AddLanguage (page 23), GetCurrentOSLanguageID (page 25), LoadLanguageMessageTable (page 26), RenameLanguage (page 117), SetCurrentOSLanguageID (page 119)

SetCurrentOSLanguageID

Sets the language ID for the language currently running on the server

Local Servers: nonblocking

Remote Servers: N/A

NetWare Server: 4.x, 5.x, 6.x

Platform: NLM

Service: Internationalization

Syntax

```
incl <nwadv.h>

int SetCurrentOSLanguageID  (
   LONG   newLanguageID);
```

Parameters

`newLanguageID`

(IN) Specifies the language ID for the new server language.

Return Values

0	Successful
1	Invalid language ID

Remarks

SetCurrentOSLanguageID changes the language running on the server to the one associated with the `newLanguageID` parameter.

See Also

AddLanguage (page 23), GetCurrentOSLanguageID (page 25), LoadLanguageMessageTable (page 26), RenameLanguage (page 117), ReturnLanguageName (page 118)

setlocale

Selects a portion of a program's locale

Local Servers: nonblocking

Remote Servers: N/A

NetWare Server: ANSI

Platform: NLM

Service: Internationalization

Syntax

```
#include <locale.h>

char *setlocale  (
   int          category,
   const char  *locale);
```

Parameters

`category`

(IN) Specifies the environment category for the specified locale.

`locale`

(IN) Points to the locale for the program.

Return Values

If the selection is successful, a string is returned to indicate the locale that was in effect before setlocale was called; otherwise, a NULL pointer is returned.

Remarks

setlocale selects a portion of a locale for a program according to the category given by the `category` parameter and the locale specified by the `locale` parameter.

A locale affects the collating sequence (the order in which characters compare with one another), the way in which certain character-handling functions operate, the decimal-point character that is used in formatted input/output and string conversion, and the format and names used in the time string produced by the strftime function.

The possible values for the argument category are as follows:

LC_ALL	Select entire environment
LC_COLLATE	Select collating sequence
LC_CTYPE	Select the character handling

LC_NUMERIC	Select the numeric-format environment
LC_TIME	Select the time-related environment

At the start of a program, the equivalent of

```
setlocale (LC_ALL, "");
```

is executed.

See Also

localeconv (page 28), NWLsetlocale (page 54), strftime (*NDK: Program Management*)

Internationalization Structures

4

This documentation alphabetically lists the Internationalization structures and describes their purpose, syntax, and fields.

- “lconv” on page 124
- “LCONV” on page 127

lconv

Holds locale information for NLM specific functions

Service: Internationalization

Defined In: nwlocale.h

Structure

```
typedef struct {
   char   decimal_point [4];
   char   thousands_sep [4];
   char   grouping [4];
   char   int_curr_symbol [8];
   char   currency_symbol [4];
   char   mon_decimal_point [4];
   char   mon_thousands_sep [4];
   char   mon_grouping [8];
   char   positive_sign [4];
   char   negative_sign [4];
   char   int_frac_digits ;
   char   frac_digits ;
   char   p_cs_precedes ;
   char   p_sep_by_space ;
   char   p_sign_posn ;
   char   n_cs_precedes ;
   char   n_sep_by_space ;
   char   n_sign_posn ;
} lconv;
```

Fields

`decimal_point`

Specifies the decimal point character used to format non-monetary quantities.

`thousands_sep`

Specifies the character used to separate groups of digits to the left of the decimal point character in non-monetary quantities.

`grouping`

Specifies the size of each group of digits in formatted non-monetary quantities.

`int_curr_symbol`

Specifies the international currency symbol applicable to the locale.

`currency_symbol`

Specifies the local currency symbol applicable to the locale.

`mon_decimal_point`

Specifies the character used as a decimal point in monetary quantities.

mon_thousands_sep

Specifies the character used to separate groups of digits to the left of the decimal point character in formatted monetary quantities.

mon_grouping

Specifies the size of each group of digits in formatted monetary quantities.

positive_sign

Specifies the character used to indicate non-negative monetary quantities.

negative_sign

Specifies the character used to indicate negative monetary quantities.

int_frac_digits

Specifies the number of fractional digits to be displayed in internationally formatted monetary quantities.

frac_digits

Specifies the number of fractional digits to be displayed in locally formatted monetary quantities

p_cs_precedes

Specifies whether the currency symbol precedes or follows the value for non-negative formatted monetary quantities:

0 Currency symbol follows
1 Currency symbol precedes

p_sep_by_space

Specifies whether the currency symbol is separated by a space from the value for non-negative formatted monetary quantities:

0 Currency symbol not separated
1 Currency symbol separated

p_sign_posn

Specifies the position of the `positive_sign` field for a formatted monetary quantity.

n_cs_precedes

Specifies whether the currency symbol precedes or follows the value for negative formatted monetary quantities:

0 Currency symbol precedes
1 Currency symbol follows

n_sep_by_space

Specifies whether the currency symbol is separated by a space from the value for negative formatted monetary quantities:

0 Currency symbol is not separated
1 Currency symbol is separated

`n_sign_posn`

Specifies the position of the `negative_sign` field for a formatted monetary quantity.

Remarks

The nwlocale.h file redefines lconv as the LCONV structure and adds additional information.

The `int_curr_symbol` structure stores the international currency symbol applicable to the current locale. The first three characters of the `int_curr_symbol` field contain the alphabetical international currency symbol in accordance with *ISO 4217 Codes for the Representation of Currency and Funds*. The fourth character (preceding the NULL terminator) is the character that separates the international currency symbol from the monetary quantity.

The elements of the `grouping` and `mon_grouping` fields are interpreted as follows:

CHAR_MAX	No further grouping is to be performed.
0	The previous element is to be repeatedly used for the remainder of the digits.
other	The value is the number of digits that comprise the current group. The next element is examined to determine the size of the nest group of digits to the left of the current group.

The values of the `p_sign_posn` and `n_sign_posn` fields are interpreted as follows:

0	Parentheses surround the quantity and currency symbol.
1	The sign string precedes the quantity and currency symbol.
2	The sign string follows the quantity and currency symbol.
3	The sign string immediately precedes the quantity and currency symbol.
4	The sign string immediately follows the quantity and currency symbol.

LCONV

Is the NetWare counterpart to the standard lconv structure

Service: Internationalization

Defined In: nwlocale.h

Structure

```
typedef struct {
   char   decimal_point [4];
   char   thousands_sep [4];
   char   grouping [4];
   char   int_curr_symbol [8];
   char   currency_symbol [4];
   char   mon_decimal_point [4];
   char   mon_thousands_sep [4];
   char   mon_grouping [8];
   char   positive_sign [4];
   char   negative_sign [4];
   char   int_frac_digits ;
   char   frac_digits ;
   char   p_cs_precedes ;
   char   p_sep_by_space ;
   char   n_cs_precedes ;
   char   n_sep_by_space ;
   char   p_sign_posn ;
   char   n_sign_posn ;
   int    code_page ;
   int    country_id ;
   char   data_list_separator ;
   char   date_separator [2];
   char   time_separator [2];
   char   time_format ;
   int    date_format ;
   char   am [meridlen];
   char   pm [meridlen];
   char   reserved [40];

} LCONV;
```

Pascal Structure

```
uses netwin32

   LCONV = packed Record
      decimal_point : Array [0..3] of Char;
      thousands_sep : Array [0..3] of Char;
      grouping : Array [0..3] of Char;
      int_curr_symbol : Array [0..7] of Char;
      currency_symbol : Array [0..3] of Char;
```

```
    mon_decimal_point : Array [0..3] of Char;
    mon_thousands_sep : Array [0..3] of Char;
    mon_grouping : Array [0..7] of Char;
    positive_sign : Array [0..3] of Char;
    negative_sign : Array [0..3] of Char;
    int_frac_digits : Char;
    frac_digits : Char;
    p_cs_precedes : Char;
    p_sep_by_space : Char;
    n_cs_precedes : Char;
    n_sep_by_space : Char;
    p_sign_posn : Char;
    n_sign_posn : Char;
    code_page : nint;
    country_id : nint;
    data_list_separator : Array [0..1] of Char;
    date_separator : Array [0..1] of Char;
    time_separator : Array [0..1] of Char;
    time_format : Char;
    date_format : Char;
    am : Array [0..MERIDLEN-1] of Char;
    pm : Array [0..MERIDLEN-1] of Char;
    reserved : Array [0..39] of Char;
End;
```

Fields

`decimal_point`

Specifies the non-monetary decimal point.

`thousands_sep`

Specifies the non-monetary separator for digits left of the decimal point.

`grouping`

Specifies a string showing the size of groups of digits.

`int_curr_symbol`

Specifies the international currency symbol for the current locale.

`currency_symbol`

Specifies the currency symbol for the current locale.

`mon_decimal_point`

Specifies the monetary decimal point.

`mon_thousands_sep`

Specifies the monetary separator for digits left of the decimal point.

`mon_grouping`

Specifies a string showing the size of groups of digits.

positive_sign

Specifies a string showing the positive monetary value.

negative_sign

Specifies a string showing the negative monetary value.

int_frac_digits

Specifies the number of fractional digits in monetary display.

frac_digits

Specifies the number of fractional digits in non-monetary display.

p_cs_precedes

Specifies the position of the currency symbol: 1=precede, 0=succeed

p_sep_by_space

Specifies whether to use a space separator with the currency symbol:

0 no space separator
1 space separator

n_cs_precedes

Specifies the location of the currency symbol for a negative monetary quantity.

n_sep_by_space

Specifies the separation of the currency symbol with a negative monetary quantity.

p_sign_posn

Specifies the value showing the position of the positive sign with a positive monetary quantity.

n_sign_posn

Specifies the value showing the position of the negative sign with a negative monetary quantity.

code_page

Specifies the code page of the local system as obtained by the NWLlocaleconv function.

country_id

Specifies the country code of the local system.

data_list_separator

Specifies the character used to separate items in a data list.

date_separator

Specifies the character used to separate month, day, and year fields in a date string.

time_separator

Specifies the character used to separate hours, minutes, and seconds in a time string.

time_format

Specifies an 8-bit code indicating the time format to be used.

`date_format`

Specifies an integer code indicating the date format to be used.

`am` (Client only)

Specifies a character string containing the local representation for ante meridian (before noon).

`pm` (Client only)

Specifies a character string containing the local representation for post meridian (after noon).

`reserved`

Is reserved.

Remarks

The first three characters of the `grouping` field contain the alphabetic international currency symbol in accordance with those specified in ISO 4217. The fourth character is the character used to separate the international currency symbol from the monetary quantity.

The first three characters of the `int_curr_symbol` field contain the alphabetic international currency symbol as defined in ISO 4217, "Codes for the Representation of Currency & Funds."

The `date_format` field has the following format:

```
char   am[MERIDLEN]
char   pm[MERIDLEN]
```

Revision History

A

The following table outlines all the changes that have been made to the Internationalization documentation (in reverse chronological order):

October 5, 2005	Transitioned to revised Novell documentation standards.
March 2, 2005	Fixed legal information.
October 6, 2004	Added documentation for NWLstrtok_r (page 81), the preferred interface in a multi-threaded, multi-processor environment.
February 18, 2004	Updated links to the *NLM User Interface Developer Components* manual, since the NWSNUT SDK has been moved to an unsupported status.
October 2002	Updated the Pascal syntax for the structures.
February 2002	Added another string example to NWprintf (page 93) in the Remarks section.
	Moved NLM Internationalization Tools chapters to *NLM User Interface Developer Components* (http://developer.novell.com/ndk/unsupported.htm#nwsnut).
	Updated links.
September 2001	Added support for NetWare 6.x to documentation.
	Added descriptions to graphics.
June 2001	Updated tables.
February 2001	Added Delphi (Pascal) syntax to functions where missing.
March 2000	Added link to Unicode in “About This Guide” on page 9.
January 2000	Removed references to the NLM platform from NWatoi (page 29), NWisalnum (page 40), NWisalpha (page 42), and NWisdigit (page 43).
	Removed "(client)" from several functions.
	Added const to syntax of input pointers.
November 1999	Added library information to each “Internationalization Functions” on page 21.
	Obsoleted NWwsprintf.
September 1999	Removed all but three options from NWLstrftime (page 64) and added a paragraph explaining the implications of the removed options.