INTL.DLL Functions Index

IntlVersion

Syntax **WORD FAR PASCAL IntlVersion(void)**

This function returns the version of the DLL. Backward compatibility is promised.

Returns Returnvalue The version number. Major version in **HIBYTE**, and minor version in **LOBYTE**. Currently 0x0100 (i.e. ver 1.0)

IntlInit

Syntax **BOOL FAR PASCAL IntlInit(LPSTR lpLanguage)**

This function initiates the DLL. It should be called before any use of the conversion functions, and when trapping a **WM_WININICHANGE** message indicating a change in the **[intl]** secion of **WIN.INI**.

Parameters	***Description***
lpLanguage	The string in which a copy of the sLanguage variable will be stored

Returns Result of initiation.
Returnvalue FALSE if initiation failed
TRUE on success.
Comment The sLanguage variable is a string containing a 3 letter abbreviation of the language to use. For example "spa" for Spanish or "itn" for Italian. Read WININI2.TXT for more information.

IntlDateToString

Syntax **LPSTR FAR PASCAL IntlDateToString(LPSTR lpString,**
int nYear,
int nMon,
int nMday,
int nWday,
int nMode)

This function converts a date, in year, month, day of month and day of week, to a string. The conversion can be either to a long or short format.

Parameters	***Description***
lpString	The string in which the result of the conversion will be stored
nYear	as in **struct tm**
nMon	as in **struct tm**
nMday	as in **struct tm** (month day)
nWday	as in **struct tm** (week day)
nMode	**IntlSHORT** or **IntlLONG**. Anything else gives unpredictable results

Returns lpString.
Comment There is no check if the date is realistic or not.
See also **IntlStringToDate**

IntlTimeToString

Syntax **LPSTR FAR PASCAL IntlTimeToString(LPSTR lpString,**
int nHour,
int nMin,
int nSec)

This function converts a time in hours, minutes and seconds, to a string.

Parameters	***Description***
lpString	The string in which the result of the conversion will be stored.
nHour	as in **struct tm**.
nMin	as in **struct tm**, or **IntlSKIP**.
nSec	as in **struct tm**, or **IntlSKIP**.

If a parameter is **IntlSKIP**, the field representing that parameter and the fields after that will be ignored.

Returns lpString.
Comment There is no check if the time is valid or not.
See also **IntlStringToTime**

IntlFloatToString

Syntax **LPSTR FAR PASCAL IntlFloatToString(LPSTR lpString,**
double dVal,
int nMode,
size_t nWidth,
int nDecimals)

This function converts a floating point number to a string. The conversion can be to any of the formats **IntlSTD**, **IntlFIX**, **IntlSCI**, **IntlENG** or **IntlPRE**.

Parameters	***Description***
lpString	The string in which the result of the conversion will be stored.
dVal	The floating point number to convert.
nMode	The conversion format to use.
nWidth	The width of the field to contain the number.
nDecimals	The number of decimals to show, or **IntlDEFAULT** to use the settings in **WIN.INI.**

Returns lpString.

See also **IntlStringToFloat** & **Floating point mode description**

IntlCurrencyToString

Syntax **LPSTR FAR PASCAL IntlCurrencyToString(LPSTR lpString,**
double dVal,
int nMode,
size_t nWidth,
int nDecimals)

This function converts a floating point number to a string representing a currency. The conversion can be to any of the formats **IntlSTD**, **IntlFIX**, **IntlSCI**, **IntlENG** or **IntlPRE**. The function is identical to **IntlFloatToString** with the addition of the currency symbols.

Parameters	***Description***
lpString	The string in which the result of the conversion will be stored.
dVal	The floating point number to convert.
nMode	The conversion format to use.
nWidth	The width of the field to contain the number.
nDecimals	The number of decimals to show, or **IntlDEFAULT** to use the settings in **WIN.INI.**

Returns lpString.
See also **IntlFloatToString**, **IntlStringToFloat** & **Floating point mode description**

IntlStringToDate

Syntax **BOOL FAR PASCAL IntlStringToDate(LPSTR lpString,**
LPINT lpnYear,
LPINT lpnMon,
LPINT lpnMday,
LPSTR FAR * lpszEnd)

This functions converts a string containing a date in short form, to year, month and day of month. The conversion is quite liberal. Either of short and long form of the year is allowed. Leading zeroes are not needed.

Parameters	***Description***
lpString	Contains the string to parse.
lpnYear	A pointer to an int, where the year (formatted as in **struct tm**) will be stored. It can also be a NULL pointer .
lpnMon	A pointer to an int, where the month (formatted as in **struct tm**) will be stored. It can also be a NULL pointer.
lpnMday	A pointer to an int, where the day of month (formatted as in **struct tm**) will be stored. It can also be a NULL pointer.
lpszEnd	A pointer to a char pointer. It will point to the location in string, where the conversion ended, either upon completion, or because of errors. You can also send a NULL pointer.

Returns Report of the success of the conversion.
Return value **TRUE** The conversion was successful.
FALSE The conversion failed

Comment There is no check if the date is valid or not.
See also **IntlDateToString**

IntlStringToTime

Syntax **BOOL FAR PASCAL IntlStringToTime(LPSTR lpString,**
LPINT lpnHour,
LPINT lpnMin,
LPINT lpnSec,
LPSTR FAR * lpszEnd)

This function converts a time, stored in a string, to hours, minutes and seconds. The conversion is quite liberal. If the am/pm equivalence is opitted, the string is interpreted as a 24 hour time. In 24 hour settig, a trailing am/pm equivalence is ignored. Leading zeroes are ignored.

Parameters	***Description***
lpString	The string containing the time.
lpnHour	A pointer to an int, where the hour (formatted as in **struct tm**) will be stored. It can also be a NULL pointer.
lpnMin	A pointer to an int, where the minute (formatted as in **struct tm**) will be stored. It can also be a NULL pointer.
lpnSec	A pointer to an int, where the second (formatted as in **struct tm**) will be stored. It can also be a NULL pointer.
lpszEnd	A pointer to a char pointer. It will point to the location in string, where the conversion ended, either upon completion, or because of errors. You can also send a NULL pointer.

Returns Report of the success of the conversion.

Return value **TRUE** The conversion was successful.
FALSE The conversion failed

Comment There is no check if the time is valid or not.
See also **IntlTimeToString**

IntlStringToFloat

Syntax **double FAR PASCAL IntlStringToFloat(LPSTR lpString,**
LPSTR FAR * lpszEnd)

This function converts string representation of a floating point number to a double. The format of the string can be any of the **IntlSTD**, **IntlFIX**, **IntlSCI**, **IntlENG** or **IntlPRE**.

Parameters	***Description***
lpString	The string containing the number to convert.
lpszEnd	A pointer to a char pointer. It will point to the location in string, where the conversion ended, either upon completion, or because of errors. You can also send a NULL pointer.

Returns The number or zero if the conversion failed.

See also **IntlFloatToString**, **IntlCurrencyToString** & **Floating point mode description**

IntlGetPhrase

Syntax **LPSTR FAR PASCAL IntlGetPhrase(LPSTR lpReturn,**
size_t nWidth,
LPSTR lpIniFile,
LPSTR lpPhrase);

This function looks up the phrase szPhrase in the current language section of the application specific ini file, specified in szIniFile, and if not found there, in INTL.INI, which contains the phrases listed in appendix E of the CUA. It the phrase is not found in neither the application specific ini file, nor INTL.INI, szPhrase will be copied into szReturn. The search is case insensitive. If the first letter of the lookup-phrase is a capital letter, the first letter of the returned string is capitalized.

Parameters	***Description***
lpReturn	Points to the buffer that will contain the returned string.
nWidth	The length of lpReturn.
lpIniFile	String containing the name of the application specific ini file. If it is either NULL or points to an empty string, only **INTL.INI** will be searched for the phrase.
lpPhrase	Points to the null terminated phrase.

Returns The pointer to lpReturn.

Comment White space and punctuation characters, according to isspace() and ispunct(), in the beginning and the end of the phrase should be omitted in the ini files. They will be automatically inserted/appended to the translated phrase.

See also **IntlGetMessage**

Example

Code example

```
IntlGetPhrase((LPSTR)szPhrase, siseof szPhrase, "myini.ini", "**
            yellow submarine **");
```

myini.ini

```
[ger]
yellow submarine=gelbes U-Boot

[swe]
yellow submarine=gul ubåt
```

IntlGetMessage

Syntax **LPSTR FAR PASCAL IntlGetMessage(LPSTR lpReturn,**
size_t nWidth,
LPSTR lpIniFile,
LPSTR lpIdent);

This function fetches a message, in the current language, defined in your application specific ini file. If the message is not found in the current language section, the [usa] section will be searched. The message identified by **szIdent** MUST be defined in the [usa] section of your application specific ini file.

Parameters	***Description***
lpReturn	Points to the buffer that will contain the returned string.
nWidth	The size of lpReturn.
lpIniFile	String containing the name of the application specific ini file.
lpIdent	Points to the null terminated identifier string of the message.

Returns The pointer to lpReturn.

Comment The message can contain the following escape sequences:

\a	Alert
\b	Backspace
\f	Form feed
\n	New line
\r	Carriage return
\t	tab
\v	Vertical tab
\'	Single quote
\"	Double quote
\\	Backslash

See also **IntlGetPhrase**

Example

Code example

```
IntlGetMessage((LPSTR)szMess,sizeof(szMess),"myini.ini",
            "Demotext");
```

myini.ini

```
[usa]
Demotext=a few words.

[swe]
Demotext=några få ord.

[ger]
Demotext=Ein par Wörter.
```

Floating point conversion modes

Mode	*Long name*	*Example*
IntlSCI	Scientific	3.45E7
IntlFIX	Fixed decimal	123,456.05
IntlSTD	Standard	43.23
IntlENG	Engineering	12.34E-6
IntlPRE	Scientific prefix	-325.1M

IntlSKIP

IntlSKIP is defined in **INTL.H** and is used when you want less than 3 fields in a time string. If the minute is **IntlSKIP** only the hour will be converted. If the second is **IntlSKIP** both hour and minute will be converted.

Date conversion modes

The macros **IntlLONG** and **IntlSHORT** are defined in **INTL.H**. They are used to select the mode of the date conversion. **IntlLONG** means converting to a long format, that most likely contains the names of the month as well as the weekday. **IntlSHORT** means the more compact format with only year, month and day of month as numbers.

Floating point conversion modes.

The macros **IntlSCI**, **IntlFIX**, **IntlSTD**, **IntlENG** and **IntlPRE** are defined in **INTL.H**

IntlSCI Scientific mode. The number will be converted to the **x.yyyEz** format.

IntlSTD The standard conversion. Numbers formatted as **xxxx.yyyy**. There are two exceptions to this. **(1)** The number is too large to be contained in the field. **(2)** The value of the number is so small, it'd be converted to **0.000**. In both cases the mode becomes **IntlSCI**.

IntlFIX Same as **IntlSTD**, except the number that are small will be displayed as **0.000** rather that converted with the **IntlSCI** mode.

IntlENG Engineering mode. Like the **IntlSCI**, but the power of 10 will always be a multiple of 3. The format can then be either **x.yyyEz**, **xx.yyEz** or **xxx.yEz**

IntlPRE Prefix mode. Like **IntlENG**, but the power of 10, will be displayed using scientific prefixes (**y, z, a**, **f**, **p**, **n**, **µ**, **m**, **k**, **M**, **G**, **T**, **P**, **E, Z or Y**) if within that range. Otherwise identical to **IntlSCI**.

LPINT is defined as a FAR pointer to int. in **INTL.H**

```
typedef int FAR *LPINT
```

CUA

Systems Application Architecture

Common User Access
Advanced Interface Design Guide

INTL.DLL  Internationalization DLL

What is the INTL.DLL?

INTL.DLL is the dynamic link library that will make your programs better fit for the international market. Not only does it contain functions to provide good international support, but using them is also as simple as not using them. As an example, the standard way to convert a floating point number to a string, is through a **sprintf**. With the DLL you'd use the function **IntlFloatToString**, which is as easy, if not even easier to use, formats the result after the international settings in **WIN.INI**f4 and provides more powerful means of chosing format than with **sprintf**. **INTL.DLL** also provides the functions **IntlGetPhrase** and **IntlGetMessage** to easily fetch words, phrases and long messages in the language currently used. Your program never have to be aware of what language is in use.

Support

As a registered user you are entitled to free technical support (I will not pay for phone-calls, but I won't charge you for contacting me) for as long as the product is alive. When the product is no longer alive, I will write you a letter stating that the product is considered dead, but support will continue 12 months after the date of the letter. The support given includes answering questions of all kinds and helping with problems, receiving bug reports. Reported bugs will be taken care of, and all users will be notified (mail-wise) that an upgrade is available. That upgrade will be free. If there is an incompatibility problem that I fail to fix, the money will be returned and all other registered users will be notified of the incompatibility. For non registered users only trivial questions and questions related to registering will be answered.

How to contact me:

Most preferred is **electronic mail**:
I check my mailbox almost every day, so this is a fast and reliable method that is independent of time differences. My E-mail address on the Internet is:
d89-bfr@sm.luth.se or
Bjorn.Fahller@ludd.luth.se

If you have access to electronic mail on any network I will try to find out how you can reach me, and notify you when confirming the registration.

Telephone:
This is definitely the fastest way. My phone number is:
+46 920-26870 (+46 920-226870 after April 11 1992).
I'm available most evenings between 6:00 PM and 10:00 PM Central European Time.
If you're calling from out of Europe, please note the time difference.

Mail:
This is very safe, but unfortunately slow. My address is:
Björn Fahller
Trollnäsvägen 3A
S-951 61 Luleå
SWEDEN

Description of INTL.INI and your application specific ini-file.

What are the .ini file for?

If you have tried the IntlDemo program, you have noticed that it is multilingual. This is done through the ini-files, which are the dictionaries of INTL.DLL. INTL.INI contains strings that are very often used, especially then the names of the months and weekdays. There are also all the strings mentioned in the CUA. INTLDEMO.INI contains the phrases and messages that are specific for the IntlDemo program. The reason for phrases and messages to be stored in ini-files, is that it makes it easy for anyone, even the end user, to add a new language to the program using the DLL. All that is necessary is to add the words in that new language, and Voila! Done. The program "speaks" that language.

That was in general terms. More into details now. The French section of INTL.INI looks like this:

```
[frn]
;French
January=janvier
Jan=janv.
February=février
Feb=fév.
March=mars
Mar=mars
April=avril
Apr=avr.
_May=mai
June=juin
Jun=juin
July=juillet
Jul=juil.
August=août
Aug=août
September=septembre
Sep=sept.
October=octobre
Oct=oct.
November=novembre
Nov=nov.
December=décembre
Dec=déc.
Sunday=dimanche
Monday=lundi
Tuesday=mardi
Wednesday=mercredi
Thursday=jeudi
Friday=vendredi
Saturday=samedi
.
.
.
```

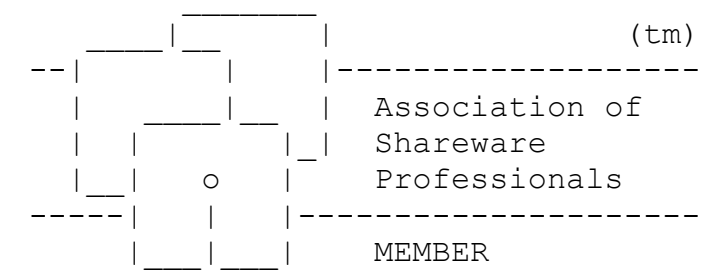
First there is a direct translation of the English names of the months, to the French names.

Also the abbrewiation of the names, are translated. If the abbrewiation of a name is identical to the 3 first letters of the name, the translation of the abbrewiation is not necessary. That is the case with May, in French. Why is may called "_May"? In some languages, the word for May is longer than 3 letters, and the abbrewiation is not necessarily the 3 first letters in that word. There must be some way of distinguishing the full name of May from it's abbrewiation. "_May" is the full name, and "May" is the abbrewiation. The same principle goes for the names of the weekdays. Other parts of INTL.INI and INTLDEMO.INI are very straightforward and self-explanatory if examined, with the possible exception of escape-codes. Long messages can contain escape codes. These are:

\a	Alert
\b	Backspace
\f	Form feed
\n	New line
\r	Carriage return
\t	tab
\v	Vertical tab
\'	Single quote
\"	Double quote
\\	Backslash

The length of a message is not limited by INTL.DLL. The upper limit (if any) is unknown, but messages with a length exceeding 650 characters are used in INTLDEMO.INI, and works.

Association of Shareware Professionals



ASP standards
ASP Ombudsman information

ASP Standards

Björn Fahller is a member of the Association of Shareware Professionals (ASP), an organization formed in April 1987 to strengthen the future of shareware as an alternative to commercial software. Its members, all of whom are programmers who subscribe to a code of ethics, are committed to the concept of shareware as a method of marketing.

The ASP's Standards for its members and their shareware products are:

Programming Standards

The program meets the ASP's definition of "shareware" (i.e.,it is not a commercial demo with a major feature disabled,nor a time-limited program).

The program has been thoroughly tested by the author and should not be harmful to other files or hardware if used properly.

Documentation Standards

Sufficient documentation is provided to allow the average user to try all the major functions of the program.

Any discussion of the shareware concept and of registration requirements is done in a professional and positive manner.

Support Standards

The member will respond to people who send registration payments, as promised in the program's documentation. At a minimum, the member will acknowledge receipt of all payments.

The member will establish a procedure for users to report, and have acknowledged, matters such as bug reports, and will describe such means in the documentation accompanying all versions of the programs. The author will respond to written bug reports from registered users when the user provides a self-addressed, stamped envelope.

Known incompatibilities with other software or hardware and major or unusual program limitations are noted in the documentation that comes with the shareware (evaluation) program.

ASP Ombudsman information

Björn Fahller is a member of the Association of Shareware Professionals (ASP). ASP wants to make sure that the shareware principle works for you. If you are unable to resolve a shareware-related problem with an ASP member by contacting the member directly, ASP may be able to help. The ASP Ombudsman can help you resolve a dispute or problem with an ASP member, but does not provide technical support for members' products. Please write to the ASP Ombudsman at

545 Grover Road,
Muskegon, MI 49442
USA

or send a CompuServe message via CompuServe Mail to ASP Ombudsman 70007,3536.

Thanks.

Very special thanks to Magnus Johansson, without whose extensive knowledge, good advice and questions, this DLL would never have been, or would at least have been quite different.

Then a big thanks to the test crew.

Magnus Johansson	Sweden	d89-mjh@sm.luth.se
Robert Heath	U.S.A	heath@ncrcae.ColumbiaSC.NCR.COM
Christian Gross	Canada	cgross@surya.uwaterloo.ca
Wolfgang Strobl	Germany	strobl@gmdzi.gmd.de
Sten Sunnergren	Sweden	stesu@sssab.se

And finally I thank you, for using it.