CLASS FILETIME64 from GWST

Implement winapi FILETIME structure + some helper methods

Structure members

union:

DWORD dwLowDateTime DWORD dwHighDateTime

ULONGLONG qft

Properties

::dDate
Date component as Xbase++ date type

::cTime
Time component as hh:mm:ss.mmm

::nTime
Time component in seconds after midnight

::cHexTs ::SetHexTs(v) ::GetHexTs()
64 bit Hexadecimal string holding the value of the filetime

::cTimeStamp
Accept Timestamp string in any of the supported formats, return timestamp in the default format YYYY-MM-DD hh:mm:ss

::nDosDateTime ::SetDosDateTime(n) ::GetDosDateTime()
DWORD holding DOS FAT date and time structures

::nExcelTime ::SetExcelTime(n) ::GetExcelTime()
64 bit floating point number holding the excel date/time format

::nUnixTime ::SetUnixTime(n) ::GetUnixTime()

64 bit integer holding the unix date/time (as Xbase++ have no support for 64 bit integers big numbers will be converted as float double)

Methods:

::SetTimeStamp(cTimeStamp [,@nShift])
::SetTimeStamp(cTimeStamp , [@nShift] , nExtendedFlags)
Set the date time using any supported timestamp string. If the timestamp include a timezone component nShift will retrieve the shift in minutes.

nExtendedFlags parameter was introduced in ot4xb on version 1.6.3.45 with the following meaning
0x01 - cTimeStamp can be also a filetime64 object
0x10 - cTimeStamp == NIL will cause ::now() applied
0x30 - cTimeStamp == NIL will cause ::nowL()applied
Flags can be combined so values 0x11 and 0x31 are also valid

::_GetTimeStamp_(cFmt)
Get the stored timestamp formatted according with the provided cFmt printf like template. DEFAULT "%04.4hu-%02.2hu-%02.2hu %02.2hu:%02.2hu:%02.2hu"

::GetTimeStamp() ::GetTimeStamp19()
Get the stored timestamp in the format **YYYY-MM-DD hh:mm:ss**

::GetTimeStamp14()
Get the stored timestamp in the format **YYYYMMDDhhmmss**

::GetIso8601()
Get the stored timestamp in the format **YYYY-MM-DDThh:mm:ss**

::SetDateTime(dDate , uTime)
Store date and time provided as Xbase++ date and time string. Time can be provided also as number of seconds after midnight.

::GetDateTime(@dDate , @cTime)
Retrieve date and time as Xbase++ date and time as 24H time string

::GetDateTime(@dDate , @nSeconds)
Retrieve date and time as Xbase++ date and number of seconds after midnight

::Now(lLocal)
Store the current timestamp, UTC or local if lLocal is provided and .T.

::ElapMilliSeconds([t2] , [lLocal])
Return the milliseconds elapsed from the stored timestamp to the provided **t2.** If t2 not provided current timestamp (UTC or local if lLocal provided and .T.) will be used to compare

::ElapSeconds([t2] , [lLocal])
Return the seconds elapsed from the stored timestamp to the provided **t2.** If t2 not provided current timestamp (UTC or local if lLocal provided and .T.) will be used to compare

::Compare(t2)
Return 0 if t2 and stored time is the same, 1 if t2 is after stored time and -1 if before.

::setRfc822(cRfc822 , @nShift)
Set the date time using an RFC822 timestamp string. If the timestamp include a timezone component nShift will retrieve the shift in minutes.

::Day() ::Month() ::Year()
Return the day of month, month or year number of the stored timestamp

::AddMilliSeconds(n) ::AddSeconds(n) ::AddMinutes(n)
::AddHours(n) ::AddDays(n) ::AddMonths(n) ::AddYears(n)
Shift the date by a positive or negative number of time units.

::ToHttp()
Return HTTP time string

::ToLocalTime()
Convert the stored UTC time to local

::strf(format)
format date time string using C strftime() from C STD library

(Introduced at ot4xb V 1.6.3.26)

NON STANDARD EXTENSION INTRODUCED AT VERSION 1.6.3.47
%S.msc will print seconds and milliseconds

FILETIME functions

Ft64_* functions are using a FILETIME structure as the first parameter, and as all structure related functions the FILETIME structure can be provided as :

- FILETIME64 object (no need to be passed by reference)
- 8 BYTE string (need to be passed by reference when the functions must modify the it) Usually initialized as ChrR(0,8)
- Numeric memory pointer to a FILETIME64 strcuture (must not passed by reference)
- Array of 2 numeric elements holding the 32 bit low and high part of the FILETIME structure.

Ft64_Add(pft,nMilliseconds ,nMultiplier)
Shift the time stored in the FILETIME structure pft by nMilliseconds * nMultiplier milliseconds

Ft64_Add_M(pft,n)
Shift the time stored in the FILETIME structure pft by n months

Ft64_Add_Y(pft,n)
Shift the time stored in the FILETIME structure pft by n years

ft64_Compare(pft,pft2[,nFlags])
Shift the FILETIME structures pft and pft2 returning 0 if equal, 1 if pft earlier than pft2 and -1 if ptf later than pft2. If pft2 is NIL will use current UTC time to compare if nFlags = 1 and current local time if nFlags = 2

ft64_ElapMilliSeconds(pft,pft2)
Return elapsed time betwen pft and pft2 in milliseconds

ft64_ElapSeconds(pft,pft2,lLocal)
Return elapsed time betwen pft and pft2 in milliseconds

Ft64_From_DosDateTime(pft,ddt) -> lOk
Initialize the pft FILETIME structure from a DWORD holding DOS FAT date and time structures

Ft64_From_EXCELTime(pft,nExcel) -> NIL
Initialize the pft FILETIME structure from a double holding Excel time

Ft64_From_UnixTime(pft,nUnix) -> NIL
Initialize the pft FILETIME structure from a DWORD holding unix time

ft64_GETDATETIME(pft,@d,@t)
Retrieve the Xbase++ date and t time string from the pft FILETIME structure

ft64_GETDATETIMESEC(pft,@d,@t)
Retrieve the Xbase++ date and t number of seconds after midnight from the pft FILETIME structure

ft64_GetTs(pft , cFmt)
Get the timestamp stored in the pft FILETIME structure formatted according with the provided cFmt printf like template. DEFAULT "%04.4hu-%02.2hu-%02.2hu %02.2hu:%02.2hu:%02.2hu"

ft64_NOW(pft,lLocal)
Store the current timestamp, UTC or local if lLocal is provided and .T.

ft64_SetDateTime(pft,d,t)
Store date and time provided as Xbase++ date and time string. Time can be provided also as number of seconds after midnight.

ft64_SetRfc822Date(pft,cStr,@nShiftInMinutes) -> NIL
Set the date time using an RFC822 timestamp string. If the timestamp include a timezone component nShift will retrieve the shift in minutes.

ft64_SetTs(pft , cStr , @nShift)
ft64_SetTs(pft , cStr , @nShift , nExtendedFlags)
Set the date time using an RFC822 timestamp string. If the timestamp include a timezone component nShift will retrieve the shift in minutes.
nExtendedFlags parameter was introduced in ot4xb on version 1.6.3.45 with the following meaning

0x01 - cTimeStamp can be also a filetime64 object
0x10 - cTimeStamp == NIL will cause ft64_NOW(@pft, .F.)
0x30 - cTimeStamp == NIL will cause ft64_NOW(@pft, .T.)
Flags can be combined so values 0x11 and 0x31 are also valid

ft64_ToHttp(pft)
Return HTTP time string

ft64_ToLocalTime(pft) -> NIL
Convert the stored UTC time to local

Ft64_To_DosDateTime(pft) -> nDosDateTime
Convert a FILETIME structure to a DWORD holding DOS FAT date and time structures

ft64_To_EXCELTime(pft) -> nExcelTime
Convert a FILETIME structure to Excel time value

ft64_To_UnixTime(pft) -> nUnixTime
Convert a FILETIME structure to unix time value

ft64_strf(@ft , cFmt , [locale_string])
format date time string using C strftime() from C STD library
(ot4xb 1.6.3.37 or higher required)

NON STANDARD EXTENSION INTRODUCED AT VERSION 1.6.3.47
%S.msc will print seconds and milliseconds

Other File and time functions

DT2ISO8601(d,t) -> cIso8601
Create an ISO 8601 string from an Xbase++ date and time string pair

FILETIME C exported functions

```
extern "C"
{
```

```
void OT4XB_API __cdecl ft64_SetTs( FILETIME* pft, LPSTR ps ,
LONG* pnShift );

LPSTR OT4XB_API __cdecl ft64_GetTs( FILETIME* pft, LPSTR pFmt);

LPSTR OT4XB_API __cdecl ft64_ToHttp( FILETIME* pft);

void OT4XB_API __cdecl ft64_AddYears( FILETIME* pft, LONG y );

void OT4XB_API __cdecl ft64_AddMonths( FILETIME* pft, LONG m );

void OT4XB_API __cdecl ft64_SetUnixTime( FILETIME* pft,
DWORD ut );

DWORD OT4XB_API __cdecl ft64_GetUnixTime( FILETIME* pft);

void OT4XB_API __cdecl ft64_SetUnixTime64( FILETIME* pft,
LONGLONG ut );

LONGLONG OT4XB_API __cdecl ft64_GetUnixTime64( FILETIME* pft);

void OT4XB_API __cdecl ft64_SetRfc822Date( FILETIME* pft,
 LPSTR ps , LONG* pnShift );

void OT4XB_API __cdecl ft64_ToLocalTime( FILETIME* pft);

LPSTR OT4XB_API __cdecl ft64_strf(FILETIME* pft, LPSTR pFormat);

LPSTR OT4XB_API __cdecl ft64_strf_l( FILETIME* pft, LPSTR pFormat
, LPSTR locale_string , int locale_category )


}
```

strftime format strings

%a Abbreviated weekday name
%A Full weekday name
%b Abbreviated month name
%B Full month name
%c Date and time representation appropriate for locale
%d Day of month as decimal number (01 - 31)
%H Hour in 24-hour format (00 - 23)
%I Hour in 12-hour format (01 - 12)
%j Day of year as decimal number (001 - 366)
%m Month as decimal number (01 - 12)
%M Minute as decimal number (00 - 59)
%p Current locale's A.M./P.M. indicator for 12-hour clock
%S Second as decimal number (00 - 59)

```
NON STANDARD EXTENSION INTRODUCED AT VERSION 1.6.3.47
  %S.msc  will print seconds and milliseconds
```

%U Week of year as decimal number, with Sunday as first day of week (00 - 53)
%w Weekday as decimal number (0 - 6; Sunday is 0)
%W Week of year as decimal number, with Monday as first day of week (00 - 53)
%x Date representation for current locale
%X Time representation for current locale
%y Year without century, as decimal number (00 - 99)
%Y Year with century, as decimal number
%z, %Z Either the time-zone name or time zone abbreviation, depending on registry settings; no characters if time zone is unknown
%% Percent sign

The # flag may prefix any formatting code.
In that case, the meaning of the format code is changed as follows.

%#a, %#A, %#b, %#B, %#p, %#X, %#z, %#Z, %#%
flag is ignored.

%#c
Long date and time representation, appropriate for current locale. For example: "Tuesday, March 14, 1995, 12:41:29".

%#x
Long date representation, appropriate to current locale. For example: "Tuesday, March 14, 1995".

%#d, %#H, %#I, %#j, %#m, %#M, %#S, %#U, %#w, %#W, %#y, %#Y
Remove leading zeros (if any).