HELP − DST woes when trying to return OLEDB Dates to a Java Client

Source: http://www.tech−archive.net/Archive/Data/microsoft.public.data.oledb/2005−12/msg00045.html

- *From*: "Bob DeRemer" <bderemer@xxxxxxxxxxxxxxxx>
- *Date*: Mon, 26 Dec 2005 07:27:02 −0800

In a nutshell, here is the problem:
* I am reading values from various data sources using our own custom, configurable OLEDB client application. It's written using VS .NET 2003 and ATL − all unmanaged code.
* The rowsets read are streamed back to a Java client application that is communicating to us via sockets − this part is fine
* We support most data types − including Dates

There is a specified assumption for the software where the data source, our OLEDB software, and the Java client software are ALL in the same TimeZone − this simplifies the whole datetime management, but we are still running into a problem.

THE PROBLEM: We need to return Date values to the Java client in the form of: number of milliseconds relative to 1, 1, 1970. The dates may be before or after 1970. For most date/time values, it works fine, but for some, the TIME part is an hour before/after [depending on whether we are in DST or not. It seems like the Win32 API(s) we are using are making adjustments on certain DST days (i.e. the sunday DST goes into effect), or in June sometimes.

Below is the function we are using to convert an incoming DATE in the 64 bit value we are trying to pass back. If you have any suggestions on how we might solve this problem, or if you see a problem in our code, please let us know.

/*
Call SystemTimeToFileTime to copy the system time to a FILETIME structure.
Call GetSystemTime to get the current system time to pass to SystemTimeToFileTime.
Copy the contents of the FILETIME structure to a ULARGE_INTEGER structure.

Initialize a SYSTEMTIME structure with the date and time of the first second of January 1, 1970.

Call SystemTimeToFileTime, passing the SYSTEMTIME structure initialized in Step 3 to the call.

Copy the contents of the FILETIME structure returned by SystemTimeToFileTime in Step 4 to a second ULARGE_INTEGER.

The copied value should be greater than or equal to the value copied in Step 2.
Subtract the 64−bit value in the ULARGE_INTEGER structure initialized in Step 2 from the 64−bit value of the ULARGE_INTEGER structure initialized in Step 5.

This produces a value in 100−nanosecond intervals since January 1, 1970. To convert this value to milliseconds, divide by 10,000.
*/
```
DATE ConvertToJavaTime(COleDateTime & dt, bool toUTC)
{
#ifdef _DEBUG
CString strTime = dt.Format();
#endif
SYSTEMTIME sysTime;
FILETIME fileTime;
__int64 realTime = 0;

if(dt.GetAsSystemTime(sysTime))
{
TzSpecificLocalTimeToSystemTime(&pv_tzi, &sysTime, &sysTime); // convert the time to UTC. (the output of this function is the number of milliseconds since 1−1−1970 UTC)
if(SystemTimeToFileTime(&sysTime, &fileTime))
{
memcpy(&realTime, &fileTime, sizeof(__int64));
}
}

// return # of milliseconds +/− since end of 1969
__int64 _outData = static_cast<__int64>((realTime − pv_zeroRealTime) / 10000); // this is UTC! we have to apply the bias to make it local.
return *reinterpret_cast<DATE*>(&_outData);
}
```

−−
Regards,

Bob DeRemer
Sr. Microsoft Architect
SAP Labs

−−
Bob DeRemer
Sr. Microsoft Architect
SAP Labs
.

- Prev by Date: ***RE: faulting module msdasql.dll Error, COM+ XA transactions***
- Next by Date: ***Re: SQLOLEDB handling Japanese data...***
- Previous by thread: ***HELP − DST woes when trying to return OLEDB Dates to a Java Client***
- Next by thread: ***CManualAccessor::CreateAccessor()***
- Index(es):
  - ***Date***
  - ***Thread***