Public Support - Support Request #1424

ADTFS-48146 Using DateTime::ToTimeStamp leads to unexpected timestamp

2017-11-30 11:30 - hidden

Status:	Closed		
Priority:	Normal		
Category:			
Requester's Priority:	Normal	**Platform:**	Ubuntu 16.04 64bit, Windows 7 64bit
Resolution:	Known Problem	**Topic:**	FileLibrary::Common
Affected Products:	ADTF File Library / IFHD 0.1.0 (BETA)	**FAQ Links:**	

Description

Supportanfrage

Hallo,

beim Ausgeben der Zeitstempel aus der ADTF File library ist uns noch etwas aufgefallen:

Verwendung der ADTF File Lib:
::a_util::datetime::DateTime dt{ ::a_util::datetime::get_current_local_date_time( ) };
auto ts = dt.ToTimeStamp( );

Hier erhalte ich sowas: 212378801678930000
Leider ist nicht klar, welche Einheit. Ich vermute aber, da in ADTF Timestamp immer Mikrosekunden hat, dass es das hier auch ist.

Verwendung der C++ Standart Lib:

C++ Standard Lib:
auto now = std::chrono::duration_cast<std::chrono::milliseconds>(std::chrono::system_clock::now( ).time_since_epoch( )).count( );

Im Vergleich dazu ist hier das Ergebnis: 1511994878930 ms
Einheit ist hier klar definiert. Das ist die Zeit seit dem 1.1.1970 (Unix Time, siehe auch https://www.epochconverter.com/ und https://en.wikipedia.org/wiki/Unix_time).

212378801678930000 / 1000 => 212378801678930

Im Vergleich:
212378801678930 ms
1511994878930 ms

Wenn ich die Ausgabe der C++ Standard Lib durch 1000 teile (um Sekunden zu erhalten) und an die DB gebe, sehe ich das korrekte Datum.
Eine Vermuting, dass der ADTF File Lib Timestamp nicht Unix Epoch ist, sondern seit dem 1.1.1900 zählt scheint auch nicht zu passen. Das wären:
3720984789 s

Könnt Ihr uns beschreiben was der Timestamp der ADTF File Lib für ein Wert ist? In der Doku haben wir dazu auch nichts gefunden.

Danke und Grüße,
Anja

Lösung

- Es liegt kein Bug vor (nur wenn man davon ausgeht, dass DateTime::ToTimeStamp einen UTC Wert bezogen auf Unixzeit liefert... was aber naheliegend wäre, wenn man sich die UTC Zeit mit Referenz zu Unix-Nullpunkt ins DateTime Objekt holt)
- Der Rückgabewert der DateTime::ToTimeStamp Methode ist Mikrosekunden
- Der Referenzpunkt ist der Nullpunkt des Julianischen Datums.
- Dazu habe ich das Snippet erweitert:

```
16  TEST(datetime_test, TestConstruction)
17  {
18      auto local_time = a_util::datetime::getCurrentLocalDateTime();
19      auto local_timestamp = local_time.toTimestamp();
20
21      auto UTC_time = a_util::datetime::getCurrentSystemDateTime();
22      auto UTC_timestamp = UTC_time.toTimestamp();
23
24      auto now = std::chrono::duration_cast<std::chrono::microseconds>(std::chrono::system_clock::now().time_since_epoch()).count();
25
26      auto diff = local_timestamp - UTC_timestamp;
27
28      auto utc_ref = a_util::datetime::DateTime(1970, 1, 1, 0, 0, 0, 0);
29      auto utc_ref_timestamp = utc_ref.toTimestamp();
30
31      auto diff_ref = UTC_timestamp - utc_ref_timestamp;
32      EXPECT_NEAR(diff_ref, now,1000);
33
```

0 %

atch 1

Name	Value
diff	3600000000
local_time	{...}
a_util::datetime::Date	{_day=5 _month=12 _year=2017 }
a_util::datetime::Time	{_hour=12 _minute=1 _second=26 ...}
local_timestamp	212379278486136000
now	1512471686136518
UTC_time	{...}
a_util::datetime::Date	{_day=5 _month=12 _year=2017 }
a_util::datetime::Time	{_hour=11 _minute=1 _second=26 ...}
UTC_timestamp	212379274886136000
utc_ref	{...}
a_util::datetime::Date	{_day=1 _month=1 _year=1970 }
a_util::datetime::Time	{_hour=0 _minute=0 _second=0 ...}
utc_ref_timestamp	210866803200000000
diff_ref	1512471686136000

siehe utc_ref_timestamp = 1. Januar 1970, 00:00 Uhr UTC

210866803200000000 us
210866803200000 ms
210866803200 s
58574112 h
240588 d
~6682 y

1970 - 6682 = -4712

- Da die DateTime::Set Methode bzw. Konstruktoren von DateTime genauso implementiert sind, kann man das DateTime Objekt wieder rekonstruieren, die Implementierung, Format und Referenz ist demnach konsistent
- Mit Hilfe des Unixzeitpunkt Referenzpunktes (siehe Snippet utc_ref_timestamp) kannst du den Wert in Unix/UTC umrechnen und abzüglich Rechenzeit entspricht das Chrono
- Wir werden in ODAUTIL-77 die Doku entsprechend erweitern

History

#1 - 2017-11-30 12:49 - hidden

- Project changed from Public Support to 7

- Status changed from New to In Progress

- Topic set to FileLibrary::Common

- Affected Products ADTF File Library / IFHD 0.1.0 (BETA) added

- Platform Ubuntu 16.04 64bit, Windows 7 64bit added

#2 - 2017-11-30 13:10 - hidden

- Status changed from In Progress to Customer Feedback Required

Hallo Anja,

da die Implementierung ja komplett offen ist (siehe ATUILS for IFHD), könnt ihr das ja auch selbst prüfen und ggf. für euren Use Case anpassen.

Ihr verwendet die **Local Time**

```
::a_util::datetime::DateTime dt{ ::a_util::datetime::get_current_local_date_time( ) };
```

Deshalb wird auch die **LocalTime** zurückgegeben, bei Windows Verwendung von GetLocalTime.

Wenn der Use Case die **System Zeit in UTC** ist, dann bitte auch entsprechend GetSystemTime verwenden.

```
::a_util::datetime::DateTime dt{ ::a_util::datetime::get_current_system_date_time( ) };
```

#3 - 2017-12-01 17:15 - hidden

Hallo Florian,

Worauf bezieht sich der Zeitstempel (welchen Startzeitpunkt)? Es ist keine Unix Time. Ist es der 1.1.0000? Ich habe gerade mit Fabian gesprochen und das ist uns immer noch nicht klar.

Grüße,
Anja

#4 - 2017-12-01 17:36 - hidden

- File convert_chrono.png added

- File time_test_snippet.JPG added

- Status changed from Customer Feedback Required to In Progress

Hi Anja,

der Rückgabewert von DateTime::ToTimestamp() *sollte* in Microsekunden sein wenn ich die Funktion anschaue (siehe Code).
Allerdings kann das so nicht stimmen, da gebe ich dir recht.
1. passt die Größenordnung nicht (Faktor 100)
2. der Wert an sich

Ich habe das ganze auch mal verglichen und bekomme folgendes Ergebnis (Hinweis wegen der Funktionsaufrufe, das ist eine neuere Version aber Code gleich, nur die Namen haben sich geändert):

```
16  TEST(datetime_test, TestConstruction)
17  {
18      auto local_time = a_util::datetime::getCurrentLocalDateTime();
19      auto local_timestamp = local_time.toTimestamp();
20
21      auto UTC_time = a_util::datetime::getCurrentSystemDateTime();
22      auto UTC_timestamp = UTC_time.toTimestamp();
23
24      auto diff = local_timestamp - UTC_timestamp;
25
26      auto now = std::chrono::duration_cast<std::chrono::microseconds>(std::chrono::system_clock::now().time_since_epoch()).count();
27
```

100 %

Watch 1

Name	Value
diff	3600000000
local_time	{...}
a_util::datetime::Date	{_day=1 _month=12 _year=2017 }
a_util::datetime::Time	{_hour=16 _minute=42 _second=19 ...}
local_timestamp	212378949739611000
now	1512142939611884
UTC_time	{...}
a_util::datetime::Date	{_day=1 _month=12 _year=2017 }
a_util::datetime::Time	{_hour=15 _minute=42 _second=19 ...}
UTC_timestamp	212378946139611000

Hier sieht man schön den Unterschied zwischen Local und UTC, die DateTime Objekte sind korrekt gefüllt, der Diff der Timestamps entspricht auch einer Stunde = Zeitzone + 1, soweit so gut.

Allerdings liefert das DateTime::ToTimestamp() scheinbar falsche Werte zurück, wie oben beschrieben.

Nimmt man den Wert der std-chrono und konvertiert diesen, entspricht es dem Wert der Aufzeichnung (vgl. DateTime Objekte):

Convert epoch to human readable date and vice versa

1512142939611000 Timestamp to Human date [batch convert timestamps to human dates]

Assuming that this timestamp is in microseconds (1/1,000,000 second):
GMT: Friday, 1. December 2017 15:42:19.611
Your time zone: Freitag, 1. Dezember 2017 16:42:19.611 GMT+01:00

Den Rückgabewert der DateTime::ToTimestamp() kann man a) gar nicht konvertieren und b) kommt beim kürzen der zwei Nullen ein Jahr 2037 usw. heraus.

Ich gebe das mal an die Utils-Entwickler weiter und melde mich dann wieder.

#5 - 2017-12-05 12:17 - hidden

- File code_snippet_datetime.JPG added

- Status changed from In Progress to Customer Feedback Required

- Resolution set to Known Problem

Hallo Anja,

ich habe das Thema ausführlich mit den Utils Entwicklern besprochen und komme zu folgenden Schluss:

- Es liegt kein Bug vor (nur wenn man davon ausgeht, dass DateTime::ToTimeStamp einen UTC Wert liefert... was aber naheliegend wäre, wenn man sich die UTC Zeit ins DateTime Objekt holt)
- Der Rückgabewert der DateTime::ToTimeStamp Methode ist Mikrosekunden
- Der Referenzpunkt ist Christi Geburt, also das Jahr 0
- Dazu habe ich das Snippet erweitert:

```
16  TEST(datetime_test, TestConstruction)
17  {
18      auto local_time = a_util::datetime::getCurrentLocalDateTime();
19      auto local_timestamp = local_time.toTimestamp();
20
21      auto UTC_time = a_util::datetime::getCurrentSystemDateTime();
22      auto UTC_timestamp = UTC_time.toTimestamp();
23
24      auto now = std::chrono::duration_cast<std::chrono::microseconds>(std::chrono::system_clock::now().time_since_epoch()).count();
25
26      auto diff = local_timestamp - UTC_timestamp;
27
28      auto utc_ref = a_util::datetime::DateTime(1970, 1, 1, 0, 0, 0, 0);
29      auto utc_ref_timestamp = utc_ref.toTimestamp();
30
31      auto diff_ref = UTC_timestamp - utc_ref_timestamp;
32      EXPECT_NEAR(diff_ref, now,1000);
33
```

0 %

atch 1

Name	Value
diff	3600000000
local_time	{...}
a_util::datetime::Date	{_day=5 _month=12 _year=2017 }
a_util::datetime::Time	{_hour=12 _minute=1 _second=26 ...}
local_timestamp	212379278486136000
now	1512471686136518
UTC_time	{...}
a_util::datetime::Date	{_day=5 _month=12 _year=2017 }
a_util::datetime::Time	{_hour=11 _minute=1 _second=26 ...}
UTC_timestamp	212379274886136000
utc_ref	{...}
a_util::datetime::Date	{_day=1 _month=1 _year=1970 }
a_util::datetime::Time	{_hour=0 _minute=0 _second=0 ...}
utc_ref_timestamp	210866803200000000
diff_ref	1512471686136000

- Da die DateTime::Set Methode bzw. Konstruktoren von DateTime genauso implementiert sind, kann man das DateTime Objekt wieder rekonstruieren, die Implementierung, Format und Referenz ist demnach konsistent
- Ich würde dennoch gerne das bestehende Ticket, dass wir 2018 angehen wollten (ACORE-8643 [Clock Service] provide option to use unixtime (UTC)), gleich umsetzen und die DateTime Funktionen so umstellen, dass sie UTC Timestamps liefern/verarbeiten können
- Damit wär ADTF 3.x / ADTF File Library auf UTC umgestellt
- Dazu müsste imho auch die File Version umgestellt werden, um nicht inkonsistent zu werden
- Mit Hilfe des UTC Referenzpunktes (siehe Snippet utc_ref_timestamp) kannst du den Wert in UTC umrechnen und abzüglich Rechenzeit

entspricht das Chrono

Geht ihr soweit mit ?
Wenn ja, darf ich dieses Ticket öffentlich machen bzw. in FAQ aufnehmen ?

#6 - 2017-12-05 14:45 - hidden

Hallo Florian,

aber irgendwie passt das mit dem Referenzpunkt Christi Geburt nicht so ganz. Wenn wir den Rückgabewert der DateTime Funktion (212378801678930000) in Jahre umrechnen kommen wir auf rund 6.7298 Jahre.

Nach etwas Recherche haben wir herausgefunden das die Differenz wohl auf Grund der Verwendung von verschiedenen Kalendersystemen zurück zu führen ist.

Warum nehmt Ihr den Startpinkt des julianischen Kalenders als Referenz (1.1.4713 v.Chr. nach gregorianischen Kalender) und nicht die Unix Time (im gregorianischem Kalender definiert)r?

Wenn Ihr die Unix Time im UTC Format angeben würdet, wäre das gut.

Wenn das Ticket abgeschlossen ist dürft ihr das Ticket öffentlich machen.

#7 - 2017-12-05 14:45 - hidden

Ich meinte natürlich julianisches Datum und nicht julianischer Kalender.

Grüße,
Anja

#8 - 2017-12-05 15:58 - hidden

Da hast du recht, die Basis ist der Nullpunkt der astronomischen Zeitskala Julianisches Datum, eben auch nochmal nachgerechnet.
Wir haben leider den Offset übersehen, der noch hinzukommt, deshalb die Annahme mit 0.

Wir werden das entsprechend dokumentieren und angeben, wie man auf Unix Zeit kommt.
Ändern werden wir es doch nicht auf Unix, da das sonst zu Inkonsistenten im DAT File führt.

Wer lieber Unix Zeit haben möchte, kann sich ja die Methoden umschreiben oder zusätzliche verwenden, der Source ist ja offen.
Es muss nur beachtet werden, dass dann die ganze Toolkette darauf angepasst werden muss.

Deshalb wäre nur die Umrechnung empfehlenswert.

Wenn keine Fragen mehr sind, würde ich das Ticket abschließen.

#9 - 2017-12-06 14:30 - hidden

Hallo Florian,

alles klar. Danke.

Dann kann das Ticket geschlossen werden.

Viele Grüße,

Anja

EB Assist ADTF Support-Team

#10 - 2017-12-07 09:01 - hidden

- Project changed from 7 to Public Support

- Subject changed from ADTFS-48146 Ausgabe DateTime der ADTF FileLib to ADTFS-48146 Using DateTime::ToTimeStamp leads to unexpected timestamp

- Description updated

- Status changed from Customer Feedback Required to To Be Closed

- Private changed from Yes to No

#11 - 2017-12-11 08:47 - hidden

- Status changed from To Be Closed to Closed

Files

convert_chrono.png	19.9 KB	2017-12-01	hidden
time_test_snippet.JPG	69.8 KB	2017-12-01	hidden
code_snippet_datetime.JPG	105 KB	2017-12-05	hidden