Public Support - Support Request #129

Copy extensions from source DAT file to target DAT file

2017-01-26 12:57 - hidden

Status:	Closed		
Priority:	Normal		
Category:			
Customer:	BOSCH	**Product Issue Numbers:**	
Department:	CC-DA/ESI	**Affected Products:**	ADTF Streaming Library 2.8.0, ADTF Streaming Library 2.9.0
Requester's Priority:	Blocker	**Platform:**	Windows 7 64bit
Support Level:	2nd Level	**Topic:**	StreamingLib::Writer
Resolution:	Solved Issue	**FAQ Links:**	

Description

Supportanfrage

Ich möchte neben den EDS-Daten auch die sog. Extensions von einem
DAT in ein anderes DAT kopieren.
Leider funktioniert es nicht richtig. Die vorhandenen Extensions,
wie z.B. "index0" bis "index4" sind ohne mein Zutun im Ziel-DAT
schon vorhanden. Ich benutze den DAT Info Dialog vom ADTF 2.13

Benutze ich die Funktionen GetExtension()/SetExtension vom
ADTFReader und ADTFWriter, so werden die bereits vorhandenen
Extensions dupliziert und es kommen die 3 fehlenden Extensions
("adtf_clock_ext...") hinzu.

Frage: Warum sind einige Extensions schon im Ziel-DAT vorhanden,
obwohl gar keine kopiert wurden? Hängt das irgendwie mit den
EDS-Dateien zusammen?

Die beiden angefügten PNGs zeigen die Extensions des Quell-DATs
und des Ziel-DATs, ohne jegliches Kopieren von Extensions.
Wie ist der Mechanismus zum Kopieren von Extensions und wie
sieht ein Code-Beispiel aus?

Lösung

- In der SDK Doku unter cADTFFile Class Reference findest du alle Extensions, die ADTF-seitig erstellt werden.
- GUID z.B. wird für jedes neue DAT File erstellt, index und index_add bezieht sich auf die Streams
- Alle weiteren Extensions müssen entweder explizit ein- oder ausgeschlossen werden
- Details siehe #129#note-14

History

#1 - 2017-01-26 15:49 - hidden

Der folgende Text wurde abgeschnitten, ab 2.13:

Ich benutze den DAT Info Dialog vom ADTF 2.13, um die Extensions zu kontrollieren.

#2 - 2017-01-26 16:16 - hidden

- Status changed from New to In Progress

- Topic set to StreamingLib::Writer

#3 - 2017-01-26 17:11 - hidden

- Status changed from In Progress to Customer Feedback Required

Hallo Gerd,

grundsätzlich werden mit der Streaming Lib gar keine Extensions kopiert.
Das muss der Anwender selbst machen.

Die von dir angesprochenen Extensions werden allerdings immer dazu geschrieben:

- Die GUID wird seit ADTF 2.12 automatisch beim Erzeugen eines DAT Files hinzugefügt bzgl. Eindeutigkeit des DAT Files
- index... und index_add... bezieht sich auf Stream (Prefix)

Diese heißen gleich, haben aber nichts mit dem Source DAT zu tun.
Hättest du dein Source File nicht mit 2.9 aufgenommen, sondern min. ADTF 2.12, wäre hier auch eine GUID.

Die clock... Extensions sind im Source DAT, weil im Recorder trace external clocks auf true war.
Die beiden EDS Extensions (Store Actual Config) weil EDS bei der Aufnahme gesetzt war.
Diese und weitere Extensions musst du manuell in der Streaming Lib kopieren, wenn du sie im Target haben willst.

Genauso wirst du das z.B. in der GUI gefragt (Create New Dat File).

Beantwortet das deine Frage ?

Hinweis an dieser Stelle (Produkttickets:

- Es gibt eine Anforderung an das Kommandozeilentool Datexporter, um auch hier nachträglich die EDS bearbeiten zu können:
  - http://km-aev.in.audi.vwg/redmine/issues/25086
- Es gibt ein Known Problem in der Streaming Lib, dass das Hinzufügen eines EDS-Directory nicht geht -> AddExtendedDataDir:
  - http://km-aev.in.audi.vwg/redmine/issues/26933
  - Workaround: Dateien müssen einzeln hinzugefügt werden -> AddExtendedData

#4 - 2017-01-27 07:26 - hidden

- File extensions.docx added

Hallo Florian,
danke für die Erklärungen. Das mit der GUID habe ich schon gehört.
Habe nochmals meine Untersuchungen in einem Word-Dokument zusammengefasst.
Das Ergebnis nach dem Kopieren der Extensions verstehe ich nicht. Wie
kann man das Duplizieren verhindern?

#5 - 2017-01-27 09:12 - hidden

Hallo Gerd,

das täuscht nun ein bisschen, diese index Extensions werden immer angelegt in einem DAT File und beziehen sich auf die Streams.
D.h. du hast die index Extensions von Source nach Target kopiert, dort sind nun die des Source und des Targets vorhanden.

#6 - 2017-01-27 09:26 - hidden

Dann ist die Frage: Wie kann ich das beim Kopieren von Source nach Target für die Streams unterbinden und nur die wirklichen Extensions kopieren?
Hier der zugehörige Code:

```
// Determine extension count of orig. DAT
    tInt nExtCount = m_datFileReader_p->GetExtensionCount();
    // Copy all the extensions from orig. to comp. DAT
    for (tInt nIdx = 0; nIdx < nExtCount; nIdx++)
{
        const cADTFFileExtension* l_FileExtension_p = nullptr;
        // Get extension of orig. DAT
        l_Result = m_datFileReader_p->GetExtension(nIdx, &l_FileExtension_p);
        if (ERR_NOERROR != l_Result)
    {
            m_outBuffer = m_outBuffer.Format("Error while getting extension <%i> of orig. DAT <%s>", nIdx, m_s
ourceDatFile.c_str());
            outputLogEntry(LOG_SOURCE_COMPRESSION_TOOL, LOG_CLASS_ERROR, m_outBuffer);
            l_Return = false;
        }

m_outBuffer = m_outBuffer.Format("File extensions <%i> = <%s>", nIdx, l_FileExtension_p->GetIdentifier());
        outputLogEntry(LOG_SOURCE_COMPRESSION_TOOL, LOG_CLASS_INFO, m_outBuffer);

// Put extension to comp. DAT
        l_Result = m_datFileWriter_p->AddExtension(l_FileExtension_p);
        if (ERR_NOERROR != l_Result)
{
            m_outBuffer = m_outBuffer.Format("Error while adding extension <%i> to comp. DAT <%s>", nIdx, m_de
stinationDatFile.c_str());
```

```
            outputLogEntry(LOG_SOURCE_COMPRESSION_TOOL, LOG_CLASS_ERROR, m_outBuffer);
            l_Return = false;
        }
    }
```

#7 - 2017-01-27 14:41 - hidden

Hallo Gerd,

in deinem Code Bsp. iterierst und kopierst du ja über alle Extensions.
Wenn du nur eine Untermenge davon haben willst (z.B. die drei Clocks), dann musst du entsprechend prüfen:

```
int nExtensionCount = pFileReader->GetExtensionCount();
const cADTFFileExtension *pExtension = NULL;
for (tInt nCounter = 0; nCounter < nExtensionCount; nCounter++)
{
    pFileReader->GetExtension(nCounter, &pExtension);
    if (NULL != pExtension)
    {
        if (0 == strcmp(pExtension->GetIdentifier(), "adtf_clock_ext_adtf"))
        {
            // Add Extension
        }
        else if (0 == strcmp(pExtension->GetIdentifier(), "adtf_clock_ext_adtf_stream"))
        {
            // Add Extension
        }
        else if (0 == strcmp(pExtension->GetIdentifier(), "adtf_clock_ext"))
        {
            // Add Extension
        }
        else
        {
            continue;
        }
    }
}
```

Oder habe ich dich nun missverstanden ?

#8 - 2017-01-30 06:58 - hidden

Hallo Florian,
das ist ja genau das Problem, ich kann ja nicht wissen, welche "Extensions" da noch in anderen DATs vorkommen, bzw. in Zukunft auftauchen. Ich will sie "nur" von einem DAT in ein neues kopieren. Es gibt sicher noch andere Extensions, von denen ich aktuell gar nichts weiß?!
Gruß Gerd

#9 - 2017-01-30 09:53 - hidden

Hallo Gerd,

OK, wenn du die Extension nur "blind" kopieren willst, dann solltest du die default Extensions ausschließen:

- Alle die mit index... beginnen, denn diese hast du bereits im Ziel DAT wegen der Streams
- Die GUID, denn diese soll bekanntlich unique für jedes DAT File sein

Dann hättest du die entscheidenden

- sämtliche Clock Extensions
- EDS (Store Actual Config und Store Actual Config 2)
- referneces files
- history

(wenn vorhanden) im Ziel DAT.

Das wäre die einzige generische Lösung die mir einfällt, ansonsten musst du das irgendwie abfragen beim User (Konsole/GUI), zur Auswahl.
Was weitere nicht per ADTF Default gelieferte Extension machen, kann ich dir natürlich so nicht sagen

#10 - 2017-01-30 10:53 - hidden

- File copied_extensions.png added

- File ext_error.png added

Hallo Florian,
ich habe jetzt nur die besagten Extensions kopiert (siehe Commandline Ausgabe)
und im Ziel sieht es dann noch merkwürdiger aus?! Wo kommen jetzt die indexN_add plötzlich her?

Habe eben festgestellt, dass man die Store_Actual_Conigs durch das Kopieren der EDS-Dateien mit in
das neue DAT transportiert?! Wie kann man die dann vom Kopieren der Extensions ausnehmen?

#11 - 2017-01-30 16:22 - hidden

Hallo Gerd,

> Wo kommen jetzt die indexN_add plötzlich her?

Ich nehme an, dass du nun mit Streaming Library 2.9 dein DAT File erstellst ?
Dann werden diese Extensions mit angelegt, dass liegt am Ringbuffer bzw. neuen DAT Format ab 2.13.x, gegen das die aktuelle Streaming Library 2.9 baut.
Das stört aber deinen Anwendungsfall nicht, du hast lediglich das Format "migriert".

> Habe eben festgestellt, dass man die Store_Actual_Conigs durch das Kopieren der EDS-Dateien mit in

das neue DAT transportiert?! Wie kann man die dann vom Kopieren der Extensions ausnehmen?

Die EDS sind diese beiden Extensions (frag bitte nicht warum die so heißen, das wird zu ADTF 3 besser bzw. hoffentlich nicht wieder "historisch gewachsen").
Du musst dich entscheiden, entweder du kopierst die Extensions mit oder du nutzt den direkten EDS (als spezielle Extension) Mechanismus, dann müsstest du die beiden Extensions beim Durchlauf/Copy allerdings wieder ausschließen.

Wie gesagt, ohne User-Interaktion sehe ich das schwierig an, gewisse Extensions zu kopieren.
Manche machen gar keinen Sinn, manche dienen der internen Nutzung, etc.
Heißt, du kannst nur explizit ausschließen oder einschließen bei einer generischen Lösung.

#12 - 2017-01-31 09:07 - hidden

Wie heißen dann die EDS-Extensions? Beginnen die immer mit "Store"?
Es gibt ja diese beiden aktuell:
Store Actual Config (ohne Underscores)
Store_Actual_Config2 (mit Underscores)

Kann ich dann alles, was mit "Store" beginnt, ausschließen?
Gibt es noch andere Namen, welche in Zukunft auftauchen können?

#13 - 2017-01-31 16:52 - hidden

Hallo Gerd,

in der SDK Doku unter cADTFFile Class Reference findest du alle Extensions, die ADTF-seitig erstellt werden.
Mich selbst irritiert der Name Store Actual Config genauso bzgl. EDS wie dich, hatte aber mal den Hintergrund, dass man in der Extended Data das aktuelle Setup (nun noch schlecht übersetzen) gespeichert hat (quasi ein Snapshot). Natürlich wurde das mal überholt, Store_Actual_Config2, schlecht gereviewed und nun hat man den Salat wegen Binärkompatibilität...
Aber das sind die beiden Extensions, die angelegt werden, wenn EDS als Extension vorhanden ist.

Der Rest kommt ebenso je nach Setup.

Das heißt diese beiden kannst du auschließen, wenn du bereits den EDS-Copy verwendest.
Ich würde dann auch explizit auf diese beiden prüfen, nicht auf StartsWith o.ä...
Oder du lässt das EDS-Copy und kopierst die beiden Extensions.

> Gibt es noch andere Namen, welche in Zukunft auftauchen können?

Stand heute wie gesagt die Liste aus der Doku.
Das ist das was von ADTF kommt.
Für die Zukunft kann ich natürlich nicht die Hand ins Feuer legen...
Aber in ADTF 2.x wird ja bekanntlich nicht mehr viel neues passieren, in ADTF 3.x ist die Thematik ohnehin anders bzw. teilweise noch offen.

Wird aber definitiv einfacher und teils auch offener.

Zu deiner Anforderung allgemein:
Ich weiß dass du es sicherlich weitesgehend automatisiert haben willst.
Ich würde eine Userinteraktion/Abfrage machen, um hier nichts "kaputt" zu machen.
Wenn deine Lösung allerdings generisch bleiben soll, dann bleibt es dir nicht erspart, Extensions entweder explizit auszuschließen oder explizit einzubinden.

#14 - 2017-02-01 15:09 - hidden

Hallo Florian,
eine Userinteraktion kommt nicht in Frage, da wir im Bereich von TB bis PB agieren!
Ich habe es jetzt so gelöst, ok aus deiner Sicht?

```
// Determine extension count of orig. DAT
    tInt nExtCount = m_datFileReader_p->GetExtensionCount();
    // Copy all the extensions from orig. to comp. DAT
    for (tInt nIdx = 0; nIdx < nExtCount; nIdx++)
{
        const cADTFFileExtension* l_FileExtension_p = nullptr;
        // Get extension of orig. DAT
        l_Result = m_datFileReader_p->GetExtension(nIdx, &l_FileExtension_p);
        if (NULL != l_FileExtension_p)
    {
            // Do not copy internally used extensions (listed at cADTFFileExtensions in ADTF SDK help)
            // There are some extensions which are used internally and can not be written by the user.
            // Those extensions are mainly used to track changes made on ADTF files. Current internal extensio
ns are:
            // - GUID
            //     Updated when writing file to hard drive using cADTFFile::Write()
            //     see cADTFFile::sstrExtensionGUID for further information
            //     for a convenient interface to query GUIDs from an ADTF file see cADTFFileGUIDExtension
            // - history
            //     Updated everytime the files content changes (Append, Remove, Replace, Clear ...)
            //     see cADTFFile::sstrExtensionHistory for further information
            //     for a convenient interface to query the history from an ADTF file see cADTFFileHistoryExten
sion
            // - index0 - index512
            //     see cADTFFile::sstrStreamIndexNamePrefix for further information
            // - index_add0 - index_add512
            //     see cADTFFile::sstrStreamIndexAdditionalNamePrefix for further information
            if (0  strncmp("index", l_FileExtension_p->GetIdentifier(), 5))
        {
                // Do NOT copy index0 .. index512
            }
            else if (0  strncmp("index_add", l_FileExtension_p->GetIdentifier(), 9))
        {
                // Do NOT copy index_add0 .. index_add512
            }
            else if (0  strcmp("history", l_FileExtension_p->GetIdentifier()))
        {
                // Do NOT copy history
            }
            else if (0  strcmp("GUID", l_FileExtension_p->GetIdentifier()))
        {
                // Do NOT copy GUID
            }
            else if (0  strcmp("Store Actual Config", l_FileExtension_p->GetIdentifier()))
        {
                // Do NOT copy EDS-Extension (will be copied automatically with EDS-Files!)
            }
            else if (0  strcmp("Store_Actual_Config2", l_FileExtension_p->GetIdentifier()))
        {
                // Do NOT copy EDS-Extension (will be copied automatically with EDS-Files!)
            }
            else
        {
                // Put extension to comp. DAT
                l_Result = m_datFileWriter_p->AddExtension(l_FileExtension_p);
                if (ERR_NOERROR != l_Result)
            {
                    m_outBuffer = m_outBuffer.Format("Error while adding extension <%s> to comp. DAT <%s>", l_
FileExtension_p->GetIdentifier(), m_destinationDatFile.c_str());
                    outputLogEntry(LOG_SOURCE_COMPRESSION_TOOL, LOG_CLASS_ERROR, m_outBuffer);
                    l_Return = false;
                }
                else
            {
                    m_outBuffer = m_outBuffer.Format("Extension <%s> of orig. DAT successfully copied to compr
essed DAT", l_FileExtension_p->GetIdentifier());
                    outputLogEntry(LOG_SOURCE_COMPRESSION_TOOL, LOG_CLASS_INFO, m_outBuffer);
                }
            }
```

```
        }
        else
    {
            m_outBuffer = m_outBuffer.Format("Error while getting extension <%i> of orig. DAT <%s>", nIdx, m_s
ourceDatFile.c_str());
            outputLogEntry(LOG_SOURCE_COMPRESSION_TOOL, LOG_CLASS_ERROR, m_outBuffer);
            l_Return = false;
        }
    }
```

#15 - 2017-02-01 17:31 - hidden

Hallo Gerd,

ja das sieht gut aus.

ich würde nach wie vor die beiden Store Actual Config Extensions (also EDS) nicht ausschließen und hier mitkopieren, dafür dann den EDS Copy weglassen.
Aber das ist lediglich eine Design Frage.

Ist das Ticket damit für dich erledigt ?
Bzw. hast du noch eine Erwartungshaltung an das Thema in diesem Ticket ?

Natürlich noch die Frage, ob das für die "Öffentlichkeit" gedacht ist mit den Code Snippets, sind zwar mehr oder weniger Example Codes etwas erweitert und keine Geheimnisse, dennoch die Nachfrage im falle des Schließens...

#16 - 2017-02-02 06:27 - hidden

Hallo Florian,
ja, Ticket ist prinzipiell erledigt.
Die EDS muss ich allerdings ausschliessen, da bei uns EDS Dateien kopiert werden müssen,
und sich damit die beiden Store Actual Configs duplizieren würden. Was das für Auswirkungen
hat, entzieht sich mir.
Ja, kann öffentlich gemacht werden!
Gruß G.

PS: Es kam hier noch die Frage auf, ob die Verlinkung von "unterteilten" DAT Sequenzen,
z.B. 6 Stück á 5 min. bei einer 30 min. Aufzeichnungsfahrt, auch mit in den Extension
stecken und kopiert werden? Ist das dann die Extension "referencedfiles"?

#17 - 2017-02-02 16:32 - hidden

- Project changed from 5 to Public Support

- Private changed from Yes to No

Hallo Gerd,

> Die EDS muss ich allerdings ausschliessen, da bei uns EDS Dateien kopiert werden müssen

Das liegt aber daran weil du die direktten EDS Methoden verwendest.
Lass diese Methoden weg und kopiere die beiden Extensions Store Actual Config und Store_Actual_Config2 mit.
Dann hast du auch EDS kopiert, das meinte ich, dass du die EDS kopieren möchtest hab ich schon richtig verstanden.
Aber beide Wege sind OK...

> Ist das dann die Extension "referencedfiles"?

Genau

Damit würde ich das Ticket abschließen ?

#18 - 2017-02-03 06:46 - hidden

Danke dir, das war neu für mich mit den Configs kopieren, das macht ja alles einfacher!
Ja, Ticket dann schließen.

#19 - 2017-02-03 08:34 - hidden

- Subject changed from Kopieren der Extensions von einem DAT in ein anderes DAT to Copy extensions from source DAT file to target DAT file

- Description updated

- Status changed from Customer Feedback Required to To Be Closed

- Resolution set to Solved Issue

#20 - 2017-02-03 08:34 - hidden

- Status changed from To Be Closed to Closed

#22 - 2017-02-21 16:21 - hidden

- Department changed from ESI to CC-DA/ESI

Files

extensions_dest.png	24.1 KB	2017-01-26	hidden
extensions_source.png	24.2 KB	2017-01-26	hidden
extensions.docx	77.8 KB	2017-01-27	hidden
copied_extensions.png	12.4 KB	2017-01-30	hidden
ext_error.png	28.8 KB	2017-01-30	hidden