Public Support - Support Request #11806

Typecast of Streams to receive data from UDP

2020-07-28 09:45 - hidden

Status:	Closed		
Priority:	Normal		
Category:			
Customer:	CARIAD	**Product Issue Numbers:**	https://www.cip.audi.de/jira/browse/ACORE-10616
Department:	Car.SW Org.	**Affected Products:**	ADTF 3.7.1
Requester's Priority:	Normal	**Platform:**	Windows 10 64bit
Support Level:	2nd Level	**Topic:**	ADTF::Common
Resolution:	Product Issue Opened	**FAQ Links:**	

Description

Supportanfrage

Gibt es eine einfache Möglichkeit, Datenströme zu "typecasten"? Bspw. erhalte ich einen anonymen Stream, welchen ich gerne live mittels DDL umcasten würde, da ich die Struktur genau kenne. Natürlich kann man ein einfaches Plugin dazu bauen. Gibt es eine einfachere bzw. offizielle Methode? Ist etwas in diese Richtung geplant?

Lösung

Hierzu gibt es nichts und ist auch nichts geplant.
ADTF 3.x sagt, ein Stream soll stets mit DDL beschrieben sein und das mitsenden.
Die einzige Ausnahme wäre ein generischer Receiver, siehe UDP Implementierung:

- https://support.digitalwerk.net/adtf/v3/adtf_html/page_demo_non_adtf_application_sender_receiver.html

Mehr ist an dieser Stelle nicht geplant.

Grundsätzlich macht es Sinn, Plugins stets mit festen Strukturen zu versehen, das kann man dann wie einen "Vertrag" sehen, gegen den man entwickelt, ausführst und funktioniert.
Alles was ungerichtet ist, ist gefährlich und v.a. im Fehlerfall nicht so einfach nachvollziehbar, gerade in komplexeren Setups, ggf. noch ohne Detailwissen der Komponenten.
Ein letzter Ansatz wäre noch ein Scripting Filter, mit dem agil in gewissen Sessions reagieren kann.

Die Streaming Source (und auch Sink) kann produktiv verwendet werden und wird supportet.
Unsere Intention war zunächst es quelloffen zu liefern, so dass die Kunden weitere Protokolle selbst anbinden können. Das haben wir mittlerweile für TCP nachgeholt und hätte auch Closed Source Basiskomponenten sein können, dann passts aber wieder nicht zur Class ID und diese können wir nicht ändern sonst laufen alte Sessions nicht...

Deshalb ist unser Verständnis von "Demos", dass wir hier den Source Code zum Verständnis und ggf. Erweiterungen liefern.

Ich habe ein Ticket (ACORE-10616) erstellt, dass aus den Labelnamen den Demo Präfix streichen soll für Filter und (Streaming/System) Services, welche eigentlich für den produktiven Einsatz gedacht sind. Das Präfix in der CID soll zeigen, dass der Source Code verfügbar ist.
Das ganze würde ich dann am Ende auch dokumentieren, um Klarheit zu schaffen.

Fazit:
Werden wir anpassen um euch mehr Sicherheit zu geben, die kannst diese Komponente ähnlich bedenkenlos einsetzen wie das Media oder Video Display

History

#1 - 2020-07-29 09:39 - hidden

- Status changed from New to In Progress

- Topic set to ADTF::Common

#2 - 2020-07-29 17:04 - hidden

- Status changed from In Progress to Customer Feedback Required

Hallo Patrick,

hierzu gibt es nichts und ist auch nichts geplant.
ADTF 3.x sagt, ein Stream soll stets mit DDL beschrieben sein und das mitsenden.
Die einzige Ausnahme wäre ein generischer Receiver, siehe UDP Implementierung:

- https://support.digitalwerk.net/adtf/v3/adtf_html/page_demo_non_adtf_application_sender_receiver.html

Mehr ist an dieser Stelle nicht geplant.

Grundsätzlich macht es Sinn, Plugins stets mit festen Strukturen zu versehen, das kann man dann wie einen "Vertrag" sehen, gegen den man entwickelt, ausführst und funktioniert.
Alles was ungerichtet ist, ist gefährlich und v.a. im Fehlerfall nicht so einfach nachvollziehbar, gerade in komplexeren Setups, ggf. noch ohne Detailwissen der Komponenten.
Ein letzter Ansatz wäre noch ein Scripting Filter, mit dem agil in gewissen Sessions reagieren kann.

Hilft dir das weiter, hast du weitere Anregungen oder macht das deinen Use Case komplizierter ?

#3 - 2020-07-30 07:58 - hidden

Um genau so einen generischen Receiver (UDP) geht es. Hier kommt ein anonymer Stream heraus. Das heißt also, daß wir einen eigenen "Typecast"-Filter bauen müssen.

Danke für die Info!

#4 - 2020-07-30 08:15 - hidden

Hi Patrick,

aber auch genau so einen UDP Filter liefern wir ja schon mit (https://support.digitalwerk.net/adtf/v3/adtf_html/page_demo_non_adtf_application_sender_receiver.html). Und der hat auch die Möglichkeit eine DDL vom Media Description Service zu holen und auf den Stream Type zu setzen.

Gibts einen Grund warum ihr nicht den nehmen könnt, bzw. warum das nicht der von Euch verwendete UDP Filter macht, der ist ja für seine Daten verantwortlich?

Grüße,

Martin

#5 - 2020-07-30 08:18 - hidden

Hi Patrick,

> Um genau so einen generischen Receiver (UDP) geht es. Hier kommt ein anonymer Stream heraus. Das heißt also, daß wir einen eigenen "Typecast"-Filter bauen müssen.

Dann solltest du eigentlich diese Streaming Source genauso verwenden können.
Anhand von Properties holt er sich die DDL vom Media Description Service und setzt den Stream Type.
Du wirst es ja genauso bereitstellen oder ?
Dann brauchst du eigentlich nichts mehr casten.

Auf Empfängerseite steht dann alles bereits im Stream, da musst du nichts mehr machen.
Das wäre die Lösung die out of the box geht.

Natürlich kannst du es auch genau anders rum machen, aber das wäre doppelter Code in meinen Augen.

#6 - 2020-07-30 08:19 - hidden

Ah, Martin hatte den gleichen Gedanken, gut dass die zwei Aussagen matchen :-)

#7 - 2020-07-30 10:20 - hidden

Das hört sich gut an. Was bedeutet "Demo" im Namen? Ist das Plugin supportet? Ist die "Demo" nur als Beispiel/Anregung für den Bau eines eigenen Filters gedacht? Wir müssen eine zuverlässige Tool-Kette aufbauen, wo solch eine Filterbezeichnung Fragen aufwerfen wird.

#8 - 2020-07-30 10:26 - hidden

Hi Patrick,

ja der ist komplett supportet und ist zusätzlich noch als Beispiel verfügbar.

@Flo: Ich denke wir müssen das "Demo" endlich aus allen Labels streichen, das führt wie wir schon öfters gesehen haben dazu, dass sie nicht verwendet werden.

Grüße,

Martin

#9 - 2020-07-30 10:52 - hidden

- Project changed from 11 to Public Support

- Subject changed from Typecast von Streams to Typecast of Streams to receive data from UDP

- Description updated

- Private changed from Yes to No

- Resolution set to Product Issue Opened

- Product Issue Numbers set to https://www.cip.audi.de/jira/browse/ACORE-10616

Hi Patrick,

wie Martin sagt, kann die Streaming Source (und auch Sink) produktiv verwendet werden und wird supportet.
Unsere Intention war zunächst es quelloffen zu liefern, so dass die Kunden weitere Protokolle selbst anbinden können. Das haben wir mittlerweile für TCP nachgeholt und hätte auch Closed Source Basiskomponenten sein können, dann passts aber wieder nicht zur Class ID und diese können wir nicht ändern sonst laufen alte Sessions nicht...

Deshalb ist unser Verständnis von "Demos", dass wir hier den Source Code zum Verständnis und ggf. Erweiterungen liefern.

Ich habe ein Ticket (ACORE-10616) erstellt, dass aus den Labelnamen den Demo Präfix streichen soll für Filter und (Streaming/System) Services, welche eigentlich für den produktiven Einsatz gedacht sind. Das Präfix in der CID soll zeigen, dass der Source Code verfügbar ist.
Das ganze würde ich dann am Ende auch dokumentieren, um Klarheit zu schaffen.

Fazit:
Werden wir anpassen um euch mehr Sicherheit zu geben, die kannst diese Komponente ähnlich bedenkenlos einsetzen wie das Media oder Video Display

#10 - 2020-07-30 10:59 - hidden

Danke für die Info. So ein Filter wäre auf jeden Fall nützlich, sofern das "Non-ADTF-Sender" im Namen so gemeint ist, daß UDP-Datagramme aus der freien Wildbahn gemeint sind, bspw. Prüfstände im Prüfgelände mit proprietären Protokollen. Meine letzte Info vor langer Zeit war, daß so ein Filter nicht geplant wäre und nur ADTF-to-ADTF-Kommunikation via UDP auf der Agenda stehen würde.

Ansonsten sind meine Fragen beantwortet. Das Ticket kann geschlossen werden.

#11 - 2020-07-30 11:08 - hidden

Hi Patrick,

> So ein Filter wäre auf jeden Fall nützlich, sofern das "Non-ADTF-Sender" im Namen so gemeint ist, daß UDP-Datagramme aus der freien Wildbahn gemeint sind, bspw. Prüfstände im Prüfgelände mit proprietären Protokollen.

Genau das ist die Intetion, du brauchst auf der Gegenseite keine IPC Implementierung, was bei ADTF-2-ADTF der Fall wäre.
Diese Seite Non-ADTF Application and IPC Integration Examples fasst das imho eigentlich gut zusammen und welche Möglichkeiten der Kommunikation und Implementierungen es gibt.

#12 - 2020-07-30 12:09 - hidden

- Customer changed from VW to Car.SW Org.

#13 - 2020-07-30 15:58 - hidden

- Status changed from Customer Feedback Required to To Be Closed

#14 - 2020-08-05 08:09 - hidden

Ich habe den Demo-Receiver nun ausprobiert: Es klappt, auch das Festlegen einer DDL für den Stream. Wenn ihr das "Demo" aus dem Namen entfernen würdet, wäre ich voll zufrieden.

Das Ticket kann geschlossen werden.

#15 - 2020-10-06 10:27 - hidden

- Status changed from To Be Closed to Closed

#16 - 2021-07-26 13:24 - hidden

- Customer changed from Car.SW Org. to CARIAD