Public Support - Support Request #1842

Documentation of SampleStream

2018-03-05 19:01 - hidden

Status:	Closed		
Priority:	Normal		
Category:			
Customer:	VW	**Product Issue Numbers:**	https://www.cip.audi.de/jira/browse/ACORE-9404; https://www.cip.audi.de/jira/browse/ACORE-9404
Department:	CARMEQ	**Affected Products:**	ADTF 3.3.0 (BETA)
Requester's Priority:	Normal	**Platform:**	Windows 10 64bit
Support Level:	2nd Level	**Topic:**	ADTF::Streaming
Resolution:	Product Issue Opened	**FAQ Links:**	

Description

Supportanfrage

Ich bin der Meinung die Doku vom Sample Stream muss erweitert werden.
(Ich bilde mir ein es mittlerweile Verstanden zu haben, aber für andere Anfänger...)

- Der ISampleStreamAccess::AsyncQueue-Modus ist zu Beginn erwähnt, aber dann doch nicht beschrieben
- Es fehlt ein deutlicher Hinweis, dass dieser Modus aber dann doch vollkommen egal ist, weil man im CE sowieso nur eine vorgefertigte Implementierung am bekommt, die keine Properties hat.
- Es fehlt der Zusammenhang mit den "asynchronen" Verbindungen im CE!!!
- Bei den Beispielfiltern wird nie Flush() aufgerufen. Welche Komponente macht das dann implizit?
  - Wenn man die Eingangsverbindung zum SampleStream auf "asynchron" setzt, dann muss man explizit Flushen.
- Wird beim Schreiben in die SampleStream-Queue eine Kopie vom Sample gemacht? Wird eine Kopie vom SampleBuffer gemacht?
- Wird beim Schreiben in die Queue eines Readers eine Kopie vom Sample oder vom SampleBuffer gemacht?
- (Wann) Ist es sinnvoll zwei SampleStreams hintereinanderzuhängen? (Das ist teilweise im Beispielprojekt so)
  - Gibt es dabei Performanceeinbußen?
- Im Allgemeinen fehlt die Begründung, warum in ADTF3 die SampleStreams überhaupt so sichtbar sind. Hätte man die nicht auch in jeden Ausgangspin (transparent) integrieren können (ähnlich zu ADTF2)? Welche Vorteile habe ich dadurch in ADTF3 konkret?

- Zusammenfassend: Die "Low-level"-Beschreibung von ISampleStream ist ja "nice-to-have". Viel wichtiger wäre aber eine Beschreibung vom Verhalten der **aktuellen** konkreten Implementierung vom SampleStream, so wie man sie im CE vorfindet und wie man dort das Verhalten ändern kann.

Lösung

> Der ISampleStreamAccess::AsyncQueue-Modus ist zu Beginn erwähnt, aber dann doch nicht beschrieben

Ja, das ist schlecht dokumentiert... wenn du dem Link folgst, siehst du immerhin, dass es dadurch der Modus des Reader umgeschalten werden kann.
Ich habe ein Doku-Ticket erstellt, wo hier nochmal nachgebessert werden muss, inkl. all deiner Anmerkungen -> ACORE-9404

> Es fehlt ein deutlicher Hinweis, dass dieser Modus aber dann doch vollkommen egal ist, weil man im CE sowieso nur eine vorgefertigte Implementierung am bekommt, die keine Properties hat.

Das hat nichts mit dem CE zu tun... in einem Reader ist nur ein Modi konfigurierbar, es nicht vorgesehen, dass dieser von außen definiert oder umgeschalten werden kann.

> Es fehlt der Zusammenhang mit den "asynchronen" Verbindungen im CE!!!

Hier gibt es keinen Zusammenhang, das eine sind (a)synchrone Reader in der Data Pipe (welche das Streaming, Samples

beeinflussen), das andere (a)synchrone Verbindungen in der Trigger Pipe (welche die Laufzeit, Trigger beeinflussen).

> Bei den Beispielfiltern wird nie Flush() aufgerufen. Welche Komponente macht das dann implizit?

Der SampleStream ruft implizit bei RunTrigger Flush auf.

> Wenn man die Eingangsverbindung zum SampleStream auf "asynchron" setzt, dann muss man explizit Flushen.

Ja, das muss noch in die Doku -> ACORE-9404

> Wird beim Schreiben in die SampleStream-Queue eine Kopie vom Sample gemacht? Wird eine Kopie vom SampleBuffer gemacht?

Nein.

> Wird beim Schreiben in die Queue eines Readers eine Kopie vom Sample oder vom SampleBuffer gemacht?

Unsere Standardimplementierung macht das nicht.

> (Wann) Ist es sinnvoll zwei SampleStreams hintereinanderzuhängen? (Das ist teilweise im Beispielprojekt so)

Das ist nicht notwendig, es ist nur entscheidend, dass ein Sample Stream zwischen einer Pin-2-Pin "Verbindung" (welche es im Konzept von ADTF 3.x ja so nicht gibt) existiert.
Wichtig wäre es aber allerdings, wenn du von einen Streaming Graphen in einen Filter Graphen verbindest und umgekehrt, da diese dann a) austauschen kannst und b) seperat voneinander hoch/runter fahren.

> Gibt es dabei Performanceeinbußen?

Ja, allerdings vernachlässigbar gering.

> Im Allgemeinen fehlt die Begründung, warum in ADTF3 die SampleStreams überhaupt so sichtbar sind. Hätte man die nicht auch in jeden Ausgangspin (transparent) integrieren können (ähnlich zu ADTF2)? Welche Vorteile habe ich dadurch in ADTF3 konkret?

Um eben den Sample Stream austauschen zu können, wenn ein Anwender/Entwickler einen eigenen hat, anderen braucht.
Woran wir noch arbeiten ist die Tool-Unterstützung, d.h. entweder ein Pop-Up wenn es mehrere gibt oder einen default Sample Stream definieren, wenn man die Verbindung wie aus ADTF 2.x gewohnt zieht -> ACORE-8373

> Zusammenfassend: Die "Low-level"-Beschreibung von ISampleStream ist ja "nice-to-have". Viel wichtiger wäre aber eine Beschreibung vom Verhalten der aktuellen konkreten Implementierung vom SampleStream, so wie man sie im CE vorfindet und wie man dort das Verhalten ändern kann.

Soll auch hiermit verbessert werden -> ACORE-9404

History

#1 - 2018-03-06 14:02 - hidden

- Status changed from New to In Progress

- Topic set to ADTF::Streaming

#2 - 2018-03-06 15:37 - hidden

- Resolution set to Product Issue Opened

- Product Issue Numbers set to https://www.cip.audi.de/jira/browse/ACORE-9404; https://www.cip.audi.de/jira/browse/ACORE-9404

Hallo Marc,

> Der ISampleStreamAccess::AsyncQueue-Modus ist zu Beginn erwähnt, aber dann doch nicht beschrieben

Ja, das ist schlecht dokumentiert... wenn du dem Link folgst, siehst du immerhin, dass es dadurch der Modus des Reader umgeschalten werden kann.
Ich habe ein Doku-Ticket erstellt, wo hier nochmal nachgebessert werden muss, inkl. all deiner Anmerkungen -> ACORE-9404

> Es fehlt ein deutlicher Hinweis, dass dieser Modus aber dann doch vollkommen egal ist, weil man im CE sowieso nur eine vorgefertigte Implementierung am bekommt, die keine Properties hat.

Das hat nichts mit dem CE zu tun... in einem Reader ist nur ein Modi konfigurierbar, es nicht vorgesehen, dass dieser von außen definiert oder umgeschalten werden kann.

> Es fehlt der Zusammenhang mit den "asynchronen" Verbindungen im CE!!!

Hier gibt es keinen Zusammenhang, das eine sind (a)synchrone Reader in der Data Pipe (welche das Streaming, Samples beeinflussen), das andere (a)synchrone Verbindungen in der Trigger Pipe (welche die Laufzeit, Trigger beeinflussen).

> Bei den Beispielfiltern wird nie Flush() aufgerufen. Welche Komponente macht das dann implizit?

Der Sache gehen wir nochmal nach, kann ich so aus dem Stegreif leider nicht beantworten.
In einem synchronen Modus geschieht das aber imho per default, im asynchronen Modus muss der Trigger explizit versendet werden, damit das Laufzeitverhalten definiert wird.

> Wenn man die Eingangsverbindung zum SampleStream auf "asynchron" setzt, dann muss man explizit Flushen.

Ja, das muss noch in die Doku -> ACORE-9404

> Wird beim Schreiben in die SampleStream-Queue eine Kopie vom Sample gemacht? Wird eine Kopie vom SampleBuffer gemacht?

Nein.

> Wird beim Schreiben in die Queue eines Readers eine Kopie vom Sample oder vom SampleBuffer gemacht?

Unsere Standardimplementierung macht das nicht.

> (Wann) Ist es sinnvoll zwei SampleStreams hintereinanderzuhängen? (Das ist teilweise im Beispielprojekt so)

Das ist nicht notwendig, es ist nur entscheidend, dass ein Sample Stream zwischen einer Pin-2-Pin "Verbindung" (welche es im Konzept von ADTF 3.x ja so nicht gibt) existiert.
Wichtig wäre es aber allerdings, wenn du von einen Streaming Graphen in einen Filter Graphen verbindest und umgekehrt, da diese dann a) austauschen kannst und b) seperat voneinander hoch/runter fahren.

> Gibt es dabei Performanceeinbußen?

Ja, allerdings vernachlässigbar gering.

> Im Allgemeinen fehlt die Begründung, warum in ADTF3 die SampleStreams überhaupt so sichtbar sind. Hätte man die nicht auch in jeden Ausgangspin (transparent) integrieren können (ähnlich zu ADTF2)? Welche Vorteile habe ich dadurch in ADTF3 konkret?

Um eben den Sample Stream austauschen zu können, wenn ein Anwender/Entwickler einen eigenen hat, anderen braucht.
Woran wir noch arbeiten ist die Tool-Unterstützung, d.h. entweder ein Pop-Up wenn es mehrere gibt oder einen default Sample Stream definieren, wenn man die Verbindung wie aus ADTF 2.x gewohnt zieht -> ACORE-8373

> Zusammenfassend: Die "Low-level"-Beschreibung von ISampleStream ist ja "nice-to-have". Viel wichtiger wäre aber eine Beschreibung vom Verhalten der aktuellen konkreten Implementierung vom SampleStream, so wie man sie im CE vorfindet und wie man dort das Verhalten ändern kann.

Soll auch hiermit verbessert werden -> ACORE-9404

@Sebastian: Kannst du dir den Punkt:

> Bei den Beispielfiltern wird nie Flush() aufgerufen. Welche Komponente macht das dann implizit?

noch einmal anschauen ?

Ansonsten sollten soweit alle Fragen beantwortet und alle Anregungen aufgenommen sein.

#3 - 2018-03-16 13:56 - hidden

Der SampleStream ruft implizit bei RunTrigger Flush auf.

Grüße,
Sebastian

#4 - 2018-03-16 14:27 - hidden

Herzlichen Dank. Ticket kann dann geschlossen werden.

#5 - 2018-03-16 15:12 - hidden

- Project changed from 20 to Public Support

- Description updated

- Status changed from In Progress to To Be Closed

- Private changed from Yes to No

#6 - 2018-03-27 16:40 - hidden

- Status changed from To Be Closed to Closed