Public Support - Support Request #10900

Cannot connect several connections on one input pin (Stream Multiplexing)

2020-03-26 11:59 - hidden

Status:	Closed		
Priority:	Normal		
Category:			
Customer:	VW	**Product Issue Numbers:**	
Department:	CARMEQ	**Affected Products:**	ADTF 3.6.3
Requester's Priority:	Normal	**Platform:**	Windows 10 64bit
Support Level:	3rd Level	**Topic:**	ADTF::Common
Resolution:	Solved Issue	**FAQ Links:**	

Description

Supportanfrage

ich habe einen Filter, der von beliebig vielen Filtern Daten entgegennehmen kann. Da die Anzahl der angeschlossenen Filter nicht festgelegt werden soll, werden alle Daten an nur einem Pin des Filters empfangen (z.B. Szenario: beliebig viele Sensoren senden Daten über das gleiche Interface und sollen fusioniert werden).

Das Anschließen von mehreren SampleStreams an einen Eingangspin ist derzeit nicht möglich. Ebenso ist das Anschließen von mehreren Ausgangspins an einen SampleStream nicht möglich.

Wie kann das oben beschriebene Verhalten mit ADTF3 realisiert werden. Falls es nicht möglich ist, ist es bereits geplant solch ein Feature in den nächsten ADTF3-Releases einzubauen?

Lösung

so einen Sample Stream wird es nicht geben. Sample Streams bilden nie eine Logik ab. Das braucht's in dem Fall aber.

Was es aber geben wird, ist ein Filter der beliebig viele Eingangspins auf einen Ausgangspin mapped. Der kann sich dann korrekt darum kümmern, dass die Daten wirklich zueinander passen und sich auch um die Trigger kümmern. Darum geht's in dem erwähnten Ticket ACORE-10391.

Wenn ihr jetzt eine Lösung braucht, dann könnt ihr folgendes ausprobieren

```
#include <adtffiltersdk/adtf_filtersdk.h>

using namespace adtf::util;
using namespace adtf::ucom;
using namespace adtf::base;
using namespace adtf::streaming;
using namespace adtf::filter;

class cStreamMerger: public cFilter
{
public:
    ADTF_CLASS_ID_NAME(cStreamMerger,
                       "stream_merger.filter.adtf.cid",
                       "Stream merger");
public:
    cStreamMerger()
    {
        m_pWriter = CreateOutputPin("output");
    }

    tResult RequestDynamicInputPin(const tChar* strName,
                                   const iobject_ptr<const IStreamType>& pType) override
    {
```

```
        CreateInputPin(strName, pType);
        RETURN_NOERROR;
    }

    tResult AcceptType(ISampleReader* /*pReader*/,
                       const iobject_ptr<const IStreamType>& pType) override
    {
        m_pWriter->ChangeType(pType);
        RETURN_NOERROR;
    }

    tResult ProcessInput(ISampleReader* /*pReader*/, const iobject_ptr<const ISample>& pSample)
override
    {
        m_pWriter->Write(pSample);
        RETURN_NOERROR;
    }

private:
    ISampleWriter* m_pWriter = nullptr;
};
```

Wie ihr seht, schert sich der gar nicht um die Typen, das heißt da müsst Ihr dann selber sehr genau drauf schauen, dass das passt.

History

#1 - 2020-03-26 14:05 - hidden

- Status changed from New to In Progress

- Topic set to ADTF::Common

#2 - 2020-03-26 14:14 - hidden

@Martin: Kannst Du das beantworten?
Betrifft dieser Use Case den kommenden SubStream Support?
Oder hilft hier das geplante Feature aus Produktticket ACORE-10391: Provide a filter to fusion several Sample Streams of same Stream Type

#3 - 2020-03-26 14:45 - hidden

Hallo Clemens,

wichtig bei der Implementation ist zu beachten, dass der Stream Type als Entität über einen Sample Stream transportiert wird (siehe: Doku). Das heißt, wenn sich der Stream Type in einem Sample Stream immer wieder ändert, muss man bei der Filter Implementation auch auf diese Änderungen reagieren (z.B. AcceptType)

Gegenfragen:

- Senden alle Filter unterschiedliche Stream Types oder alle den gleichen?
- Gibt es einen Grund dafür, dass alle Samples auf einem Input Pin empfangen werden müssen?
- Kennst du die Funktionalität der "Dynamic Pins" in ADTF?

Dynamic Pins

Hiermit kann man beliebig viele Pins von beliebigen Typen im Configuration Editor mit z.B. einem Filter verbinden. Im Code kann man dann zur Laufzeit via RequestDynamicInputPin oder RequestDynamicOutputPin pro angefragtem/verbundenem Pin darauf reagieren und entsprechend eine der CreateInputPin(...) oder CreateOutputPin() Funktionen aufrufen.

#4 - 2020-03-26 15:17 - hidden

Hallo Rick,

1.: Alle Eingangsdaten sind vom gleichen Typ. Es muss also kein AcceptType() überschrieben werden.
2.: Das spart in unserer Konfiguration das Konfigurieren von sehr vielen Verbindungen. So müssen alle Verbindungen bis zum Filter durchgeschleift werden und das durch mehrere Sub-Graphen. In ADTF2 hatte uns das an vielen Stellen den Konfigurationsaufwand erleichtert.
3. "Dynamic Pins" kennen wir. Das wäre die Alternative, um beliebige Eingangspins zu konfigurieren.

Wäre alternativ ein erweiterter SampleStream denkbar, auf dessen Sample-Queue mehrere Writer schreiben können? Ähnlich wie das folgendes Darstellung suggeriert: Sample Stream

#5 - 2020-03-26 16:13 - hidden

Hi zusammen,

nein so einen Sample Stream wird es nicht geben. Sample Streams bilden nie eine Logik ab. Das braucht's in dem Fall aber.

Was es aber geben wird, ist ein Filter der beliebig viele Eingangspins auf einen Ausgangspin mapped. Der kann sich dann korrekt darum kümmern, dass die Daten wirklich zueinander passen und sich auch um die Trigger kümmern. Darum geht's in dem erwähnten Ticket ACORE-10391.

Wenn ihr jetzt eine Lösung braucht, dann könnt ihr folgendes ausprobieren

```
#include <adtffiltersdk/adtf_filtersdk.h>

using namespace adtf::util;
using namespace adtf::ucom;
using namespace adtf::base;
using namespace adtf::streaming;
using namespace adtf::filter;

class cStreamMerger: public cFilter
{
public:
    ADTF_CLASS_ID_NAME(cStreamMerger,
                       "stream_merger.filter.adtf.cid",
                       "Stream merger");
public:
    cStreamMerger()
    {
        m_pWriter = CreateOutputPin("output");
    }

    tResult RequestDynamicInputPin(const tChar* strName,
                                   const iobject_ptr<const IStreamType>& pType) override
    {
        CreateInputPin(strName, pType);
        RETURN_NOERROR;
    }

    tResult AcceptType(ISampleReader* /*pReader*/,
                       const iobject_ptr<const IStreamType>& pType) override
    {
        m_pWriter->ChangeType(pType);
        RETURN_NOERROR;
    }

    tResult ProcessInput(ISampleReader* /*pReader*/, const iobject_ptr<const ISample>& pSample) override
    {
        m_pWriter->Write(pSample);
        RETURN_NOERROR;
    }

private:
    ISampleWriter* m_pWriter = nullptr;
};
```

Wie ihr seht, schert sich der gar nicht um die Typen, das heißt da müsst Ihr dann selber sehr genau drauf schauen, dass das passt.

Hoffe das hilft fürs erste,

Grüße,

Martin

#6 - 2020-03-26 16:15 - hidden

Wenn ihr was Ähnliches für unterschiedliche Daten haben wollt, seht Euch bitte die Substreams in ADTF 3.7 an.

#7 - 2020-03-26 16:34 - hidden

Hallo Martin,

danke für deine Antwort. Die Aussage ist auf jeden Fall eindeutig.
Der Filter hilft auf jedem Fall bei genau dem Problem.

Vielen Dank.
Damit kann das Ticket dann zu.

#8 - 2020-03-26 16:40 - hidden

- Status changed from In Progress to To Be Closed

- Resolution set to Solved Issue

- Platform Windows 10 64bit added

Super

#11 - 2020-07-07 16:35 - hidden

- Project changed from 20 to Public Support

- Subject changed from Mehrfachverbindungen an Eingangspins nicht möglich to Cannot connect several connections on one input pin (Stream Multiplexing)

- Description updated

- Private changed from Yes to No

- Support Level changed from 2nd Level to 3rd Level

#12 - 2020-07-07 16:41 - hidden

- Status changed from To Be Closed to Closed