

Public Support - Support Request #11752

Error target pin can not have multiple sources

2020-07-20 14:17 - hidden

Status:	Closed		
Priority:	Normal		
Category:			
Customer:	AUDI	Product Issue Numbers:	https://www.cip.audi.de/jira/browse/ACORE-10619
Department:		Affected Products:	ADTF 3.7.1
Requester's Priority:	Normal	Platform:	Windows 10 64bit
Support Level:	3rd Level	Topic:	ADTF::Streaming
Resolution:	Product Issue Opened	FAQ Links:	

Description

Supportanfrage

When trying to connect more than one source to an input pin in the configuration editor, I get the error:
"target pin can not have multiple sources.
Continue anyway Yes|No"
I wanted to ask, why isn't this possible in adtf3? what should be the right way doing this?
What will happen if I pressed Yes and continued?

Lösung

I wanted to ask, why isn't this possible in adtf3?

Only 1:1 connections are valid for pins.
The only exception is Sample Stream to Input Pin (1:N) to bring same data to different receiver.
But N:1 connections are never allowed.

Especially I don't see any use case for transferring multiple (different) data to one defined streaming input.

What will happen if I pressed Yes and continued?

Then it will be marked as invalid connections. It is possible for the use case connecting more than one incoming connection but only activate a single one and deactivate the other.
Then you can toggle for different launches.

I am sure that the users will start implementing their own filters to merge streams and forward them on one output pin.

we currently do not have such a filter, but in the mean time you can use the following:

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

class cSampleStreamMerger: public adtf::filter::cFilter
{
public:
    ADTF_CLASS_ID_NAME (cSampleStreamMerger,
                        "sample_stream_merger.filter.me",
                        "Sample Stream Merger");

    cSampleStreamMerger()
    {
        m_pWriter = CreateOutputPin("output");
    }

    tResult RequestDynamicInputPin(const tChar* strName,
```

```

                                const adtf::ucom::iobject_ptr<const adtf::streaming::
IStreamType>& pType)
    {
        CreateInputPin(strName, pType);
        RETURN_NOERROR;
    }

    tResult AcceptType(adtf::streaming::ISampleReader* /*pReader*/,
                        const adtf::ucom::iobject_ptr<const adtf::streaming::IStreamType>& pType)
    {
        if (m_pStreamType)
        {
            RETURN_IF_FAILED(adtf::streaming::is_compatible(pType, m_pStreamType));
        }

        m_pWriter->ChangeType(pType);
        m_pStreamType = pType;
        RETURN_NOERROR;
    }

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

private:
    adtf::streaming::ISampleWriter* m_pWriter = nullptr;
    adtf::ucom::object_ptr<const adtf::streaming::IStreamType> m_pStreamType;
};

```

I have created ticket ACORE-10619 to include the above filter into the standard ADTF distribution.

History

#1 - 2020-07-20 17:46 - hidden

- Status changed from New to In Progress
- Topic set to ADTF::Streaming
- Customer set to AUDI

#2 - 2020-07-23 18:56 - hidden

- Status changed from In Progress to Customer Feedback Required
- Resolution set to Not Supported Scope

Hi Samer,

I wanted to ask, why isn't this possible in adtf3?

Only 1:1 connections are valid for pins.
The only exception is Sample Stream to Input Pin (1:N) to bring same data to different receiver.
But N:1 connections are never allowed.

Especially I don't see any use case for transferring multiple (different) data to one defined streaming input.

What will happen if I pressed Yes and continued?

Then it will be marked as invalid connections. It is possible for the use case connecting more than one incoming connection but only activate a single one and deactivate the other.
Then you can toggle for different launches.

#3 - 2020-07-29 17:28 - hidden

- Description updated

- Status changed from Customer Feedback Required to To Be Closed

- Private changed from Yes to No

#4 - 2020-07-30 18:48 - hidden

The data we want to pass to the input port are not different.

There are some use cases which this feature is required. For example filters with sample queue on their inputs; sorting the coming data in a specific way.

I know that you can do this with multiple or dynamic pins. The thing is; I am migrating ADTF2 filters to ADTF3. This missing feature will force me to change the logic of the pins of the migrated filters. Which adds more complexity to the migration.

I am sure that the users will start implementing their own filters to merge streams and forward them on one output pin.

Do the "Substream Assembler" or "Substream Merge" have something to do with this feature?

#5 - 2020-07-30 21:38 - hidden

- Status changed from To Be Closed to In Progress

#6 - 2020-08-05 09:49 - hidden

- Resolution changed from Not Supported Scope to Product Issue Opened

- Product Issue Numbers set to <https://www.cip.audi.de/jira/browse/ACORE-10619>

Hi Samer,

we currently do not have such a filter, but in the mean time you can use the following:

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

class cSampleStreamMerger: public adtf::filter::cFilter
{
public:
    ADTF_CLASS_ID_NAME(cSampleStreamMerger,
                      "sample_stream_merger.filter.me",
                      "Sample Stream Merger");

    cSampleStreamMerger()
    {
        m_pWriter = CreateOutputPin("output");
    }

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

    tResult AcceptType(adtf::streaming::ISampleReader* /*pReader*/,
                      const adtf::ucom::iobject_ptr<const adtf::streaming::IStreamType>& pType)
    {
        if (m_pStreamType)
        {
            RETURN_IF_FAILED(adtf::streaming::is_compatible(pType, m_pStreamType));
        }

        m_pWriter->ChangeType(pType);
        m_pStreamType = pType;
        RETURN_NOERROR;
    }

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

private:
    adtf::streaming::ISampleWriter* m_pWriter = nullptr;
    adtf::ucom::object_ptr<const adtf::streaming::IStreamType> m_pStreamType;
};
```

I have created ticket <https://www.cip.audi.de/jira/browse/ACORE-10619> to include the above filter into the standard ADTF distribution.

Regards,

Martin

#7 - 2020-08-10 16:50 - hidden

- *Description updated*
- *Status changed from In Progress to To Be Closed*
- *Support Level changed from 2nd Level to 3rd Level*

#8 - 2020-10-06 10:27 - hidden

- *Status changed from To Be Closed to Closed*