Public Support - Support Request #11155

Using dynamic properties in ADTF 3.x

2020-05-05 12:48 - hidden

Status:	Closed		
Priority:	Normal		
Category:			
Customer:	ACADEMIC	**Product Issue Numbers:**	https://www.cip.audi.de/jira/browse/ACORE-10403; https://www.cip.audi.de/jira/browse/ACORE-10536
Department:	LINZ	**Affected Products:**	ADTF 3.6.3
Requester's Priority:	Normal	**Platform:**	Ubuntu 16.04 64bit
Support Level:	2nd Level	**Topic:**	ADTF::CE
Resolution:	Solved Issue	**FAQ Links:**	

Description

Supportanfrage

ich würde gerne bei einem Streaming Source Filter dynamische Properties hinzufügen.

Im Detail: Man setzt im Configuration Editor z.B.: ein existierendes Property "Anzahl Felder" auf den Wert 3.
Danach sollten die Properties Feld0, Feld1, Feld2 dynamisch erzeugt werden, in Editor unter Properties angezeigt werden und Werte zugewiesen werden können.
Das Ganze sollte noch vor der eigentlichen Initalisierungsphase des Filters passieren.

Gibt es dazu eine Möglichkeit?

Lösung

Das wird ab ADTF 3.7.0 unterstützt

Um dynamische Elemente wie Pins und Properties zu erzeugen, wird in ADTF 3.x ein QML Filter Editor verwendet.
D.h. du musst in deinen Filter Code im Konstruktor ein Qml File relativ zur Plugindescription referenzieren (siehe SetEditor).

Dann bekommt dein Filter ein Kontextmenu Eintrag und ein Qml Code wird ausgeführt.
Hierzu gibt es eine API und Bsp, siehe https://support.digitalwerk.net/adtf/v3/adtf_html/page_filter_editor.html
Der Player nutzt z.b. diesen Mechanismus um Pins für die Streams aus dem adtfdat file anzulegen, das qml file findest du ebenso im delivery.

Eine Property Variable bietet dir quasi eine Referenz auf eine statische Variable an.
Das klappt natürlich bei dynamischen Properties nicht.
Du kannst direkt auf der Property arbeiten, wenn es keine statische ist.

Im folgenden Snippet hast du z.B. eine Property myFilename vom Typ cFilename erstellt:

```
// get all properties
object_ptr<const IProperties> pProperties;
RETURN_IF_FAILED(this->GetProperties(pProperties));

// get the specific property
adtf::base::ant::property<cFilename> oProp;
RETURN_IF_FAILED(pProperties->GetProperty("myFilename", oProp));

// do something, e.g. print
cString strValue;
oProp.GetValue()->ToString(adtf_string_intf(strValue));
LOG_INFO(strValue.GetPtr());
```

Alternativ kannst auch ausschließlich statt dem Snippet die elegantere template Funktion verwendet werden:

```
auto strFilename = adtf::base::get_property<cFilename>(*this, "myFilename");
```

Um Auch Sub-Properties auszulesen nimmt man am besten

```
auto strFilename = adtf::base::get_property_by_path<cFilename>(*this, "parent/child/myFilename");
```

Zur Vollständig:

Man kann property_variable<> auch für "dynamische" Properties verwenden, müssen dann aber an anderer Stelle registriert werden um nicht in der Plugin Description zu landen:

```
// zur Aufbewahrung ein Member
std::vector<std::unique_ptr<property_variable<tInt32>>> m_oProperties;

// dann in der Init oder sonst wo
for (tSize nPropertyIndex = 0; nPropertyIndex < 10; ++nPropertyIndex)
{
    m_oProperties.push_back(std::make_unique<property_variable<tInt32>>());
    RegisterPropertyVariable(("value" + std::to_string(nPropertyIndex)).c_str(), *m_oProperties.
back());
}
```

Ich empfehle aber den vereinfachten template Weg zum Einlesen und einen QML Filter Editor zum Anlegen im CE.

Related issues:

Related to Public Support - Support Request #11244: Using dynamic properties ...	**Closed**
Related to Public Support - Support Request #11587: Dynamic property is not c...	**Closed**
Related to Public Support - Support Request #11623: Generate dynamic properties	**Closed**

History

#1 - 2020-05-06 08:44 - hidden

- Status changed from New to Customer Feedback Required

- Topic set to ADTF::CE

Hallo Patrick,

ja das geht.
Um dynamische Elemente wie Pins und Properties zu erzeugen, wird in ADTF 3.x ein QML Filter Editor verwendet.
D.h. du musst in deinen Filter Code im Konstruktor ein Qml File relativ zur Plugindescription referenzieren (siehe SetEditor).
Dann bekommt dein Filter ein Kontextmenu Eintrag und ein Qml Code wird ausgeführt.
Hierzu gibt es eine API und Bsp, siehe https://support.digitalwerk.net/adtf/v3/adtf_html/page_filter_editor.html

Der Player nutzt z.b. diesen Mechanismus um Pins fr die Streams aus dem adtfdat file anzulegen, das qml file findest du ebenso im delivery.

#2 - 2020-05-06 20:20 - hidden

Hallo Florian,

danke erstmal.
Das SetEditor -Beispiel mit einem in der Filter-Klasse vorhandenen, registrierten Property, welches dann über die QML GUI verändert werden kann, hat super funktioniert.

Jedoch habe ich immer noch Probleme mit dem Erstellen von neuen Properties. Das createProperty -Beispiel hat nicht funktioniert. Verstehe ich es richtig, dass nach Eingabe von validen Daten und nach einem Klick auf den QML-Button, das Property im **Editor** rechts unter **Properties** erscheinen sollte? Und wie kann ich danach innerhalb meines Filters diese Properties zum Beispiel bei der überschriebenen Init-Methode verwenden, wenn ich keine private Membervariable für das Property habe?

Danke für die Hilfe.
Patrick

#3 - 2020-05-07 14:01 - hidden

Hallo Patrick,

> Verstehe ich es richtig, dass nach Eingabe von validen Daten und nach einem Klick auf den QML-Button, das Property im Editor rechts unter Properties erscheinen sollte?

Ich war leider meiner Zeit voraus, das Feature ist ganz frisch und kommt mit der 3.7.0 (wird im Laufe des Tages released).

Die Pakete sind aber schon verfügbar, sowohl im Download Archive als auch im Artifactory/conan:

- https://support.digitalwerk.net/projects/download-center/repository/show/adtf/release-3.7.0
- ADTF/3.7.0@dw/stable -> http://artifactory.digitalwerk.lan:8081/artifactory/webapp/#/artifacts/browse/tree/General/dw-prod-conan-releases/dw/ADTF/3.7.0/stable/package

  Und wie kann ich danach innerhalb meines Filters diese Properties zum Beispiel bei der überschriebenen Init-Methode verwenden, wenn ich keine private Membervariable für das Property habe?

Eine Property Variable bietet dir quasi eine Referenz auf eine statische Variable an.
Das klappt natürlich bei dynamischen Properties nicht.
Du kannst direkt auf der Property arbeiten, wenn es keine statische ist.

Im folgenden Snippet hast du z.B. eine Property myFilename vom Typ cFilename erstellt:

```
// get all properties
object_ptr<const IProperties> pProperties;
RETURN_IF_FAILED(this->GetProperties(pProperties));

// get the specific property
adtf::base::ant::property<cFilename> oProp;
RETURN_IF_FAILED(pProperties->GetProperty("myFilename", oProp));

// do something, e.g. print
cString strValue;
oProp.GetValue()->ToString(adtf_string_intf(strValue));
LOG_INFO(strValue.GetPtr());
```

#4 - 2020-05-07 18:30 - hidden

Hallo Florian,

mit der Version 3.7.0 funktioniert nun alles und dein Snippet hilft mir sehr bei meiner Implementierung.

Besten Dank!
Patrick

#5 - 2020-05-08 09:05 - hidden

- Subject changed from Dynamische Properties to Using dynamic properties in ADTF 3.x

- Description updated

- Status changed from Customer Feedback Required to To Be Closed

- Private changed from Yes to No

- Resolution set to Solved Issue

Wunderbar !

#6 - 2020-05-14 13:02 - hidden

- Related to Support Request #11244: Using dynamic properties in ADTF 3.x added

#7 - 2020-05-19 08:10 - hidden

Hallo Patrick,

für das Snippet von Flo, gibt es auch einen template Funktion die genau das alles vereint:

```
auto strFilename = adtf::base::get_property<cFilename>(*this, "myFilename");
```

Grüße,

Martin

#8 - 2020-05-19 08:16 - hidden

Der Vollständigkeit halber: Man kann property_variable<> auch für "dynamische" Properties verwenden:

```
// zur Aufbewahrung ein Member
std::vector<std::unique_ptr<property_variable<tInt32>>> m_oProperties;

// dann in der Init oder sonst wo
```

```
for (tSize nPropertyIndex = 0; nPropertyIndex < 10; ++nPropertyIndex)
{
    m_oProperties.push_back(std::make_unique<property_variable<tInt32>>());
    RegisterPropertyVariable(("value" + std::to_string(nPropertyIndex)).c_str(), *m_oProperties.back());
}
```

#9 - 2020-05-19 08:22 - hidden

- Description updated

#10 - 2020-05-19 08:58 - hidden

Hallo Martin,

danke für den zusätzlichen Hinweis!
In der Zwischenzeit hat sich bei mir eine neue Frage aufgetan. :)

Können dynamische Properties, die über die QML-API mit

`createProperty(...)`

erzeugt wurden, auch hidden oder read-only sein?
Somit könnte man schon eine erste Validierung des Inputs innerhalb der QML GUI durchführen, da anschließend diese im ADTF Config Editor unter Properties nicht mehr verändert werden können. Für meine Anwendung wäre es von Vorteil, wenn der User **immer** die Konfiguration über den QML Filter Editor vornimmt. Nach dem Speichern sind diese schreibgeschützt im ADTF Config Editor unter Properties sichtbar. Ist das derzeit möglich ?

Patrick

#11 - 2020-05-22 08:46 - hidden

Hallo nochmal,

außerdem werde ich nicht schlau daraus, wieso beim Aufruf von createProperty(...) über die QML-API immer eine MessageBox mit der Info "Argument contains a null pointer" erscheint.
Das Property wird dennoch mit dem richtigen Typ, Namen und Wert beim korrekten Filter hinzugefügt. targetModel kann somit auch kein nullpointer sein.

Patrick

#12 - 2020-05-25 17:18 - hidden

- Status changed from To Be Closed to In Progress

#14 - 2020-05-29 16:58 - hidden

- Status changed from In Progress to Customer Feedback Required

- Product Issue Numbers set to https://www.cip.audi.de/jira/browse/ACORE-10403

Hallo Patrick,

> Können dynamische Properties, die über die QML-API mit
>
> `createProperty(...)`
>
> erzeugt wurden, auch hidden oder read-only sein?
> Somit könnte man schon eine erste Validierung des Inputs innerhalb der QML GUI durchführen, da anschließend diese im ADTF Config Editor unter Properties nicht mehr verändert werden können. Für meine Anwendung wäre es von Vorteil, wenn der User **immer** die Konfiguration über den QML Filter Editor vornimmt. Nach dem Speichern sind diese schreibgeschützt im ADTF Config Editor unter Properties sichtbar. Ist das derzeit möglich ?

Hidden Dynamic Properties machen für mich keinen Sinn, weil du ja den Anwender dazu bewegst, Properties anzulegen/zu bedaten und dann verstecken ? Das spiest sich irgendwie, ggf. ist dann auch ein Property der falsche Weg oder ich verstehe den Use Case nicht.

Read-Only Properties gibt es derzeit nicht in ADTF 3.x.
Insgesamt kann man zu jederzeit eine Propery einer Komponente setzen, die Frage ist, ob diese auch darauf reagiert.
Eine Anforderung für Read-Only Property besteht aber bereits, muss allderings noch umgesetzt werden und ist bis Jahresende geplant (ACORE-10403).

> außerdem werde ich nicht schlau daraus, wieso beim Aufruf von createProperty(...) über die QML-API immer eine MessageBox mit der Info "Argument contains a null pointer" erscheint.
> Das Property wird dennoch mit dem richtigen Typ, Namen und Wert beim korrekten Filter hinzugefügt. targetModel kann somit auch kein nullpointer sein.

Das ist ein Bug im QML/CE bzw. dem Script... das tritt aktuell bei uns auch auf und liegt nicht an deinem Doing.
Deshalb bitte ignorieren, das muss noch debugged und gefixt werden (ACORE-10536).

#15 - 2020-05-29 17:02 - hidden

- Product Issue Numbers changed from https://www.cip.audi.de/jira/browse/ACORE-10403 to https://www.cip.audi.de/jira/browse/ACORE-10403; https://www.cip.audi.de/jira/browse/ACORE-10536

#16 - 2020-06-01 21:35 - hidden

Hallo Florian,

> Hidden Dynamic Properties machen für mich keinen Sinn, weil du ja den Anwender dazu bewegst, Properties anzulegen/zu bedaten und dann verstecken ? Das spiest sich irgendwie, ggf. ist dann auch ein Property der falsche Weg oder ich verstehe den Use Case nicht.

Mein konkreter Usecase ist ein Sensor, der mit einer Streaming Source implementiert wird. Für die Konfiguration werden dynamische Properties benötigt, da gewisse Unterstrukturen der Gesamt-Konfiguration mehrmals hinzugefügt werden können/müssen. Somit ist der einzige Weg über den QML Editor selbst. Hier werden diese Properties angelegt und später bei der Initalisierung der Streaming Source abgegriffen und der Sensor konfiguriert. Bei der QML GUI kann bereits ein Validierung der konfigurierten Werte/Datentypen erfolgen. Später können diese Werte im Konfigurations-Editor ohne der QML GUI jedoch wieder zu invaliden Daten verändert werden. Somit wäre es für meinen Usecase von Vorteil, wenn es Read-Only oder eben Hidden Properties geben würde. So wird der Verwender der Streaming Source quasi gezwungen die Konfiguration immer über die QML GUI vorzunehmen. Es funktioniert natürlich trotzdem ohne diesem "Feature", aber vielleicht hilft euch dieses Feedback.

> Das ist ein Bug im QML/CE bzw. dem Script... das tritt aktuell bei uns auch auf und liegt nicht an deinem Doing.
> Deshalb bitte ignorieren, das muss noch debugged und gefixt werden (ACORE-10536).

Alles klar, dann werde ich diese Meldung zur Zeit ignorieren.
Danke für die Hilfe!

Patrick

#17 - 2020-06-03 08:35 - hidden

Hi Patrick,

wenn die Properties eigentlich nicht direkt dafür gedacht sind, vom User manipuliert zu werden, dann geht auch ihr Vorteil durch den generischen Property Editor verloren. Daher wäre es für Deinen Use Case vielleicht auch eine Möglichkeit auf eine Konfigurationsdatei, die vom QML Editor erzeugt wird und dann von der Streaming Source interpretiert wird, umzusteigen. Das ist zwar dann eine Datei mehr, aber die kann man dann auch komfortabel für neue Sessions wiederverwenden. Properties passen natürlich trotzdem, nur so als Denkanstoß.

Grüße,

Martin

#18 - 2020-06-06 16:03 - hidden

Hi Martin,

das geht natürlich auch - danke!

Beste Grüße,
Patrick

#19 - 2020-06-08 08:24 - hidden

- Status changed from Customer Feedback Required to To Be Closed

#20 - 2020-06-29 08:20 - hidden

- Related to Support Request #11587: Dynamic property is not created added

#21 - 2020-07-03 15:03 - hidden

- Related to Support Request #11623: Generate dynamic properties added

#22 - 2020-07-07 12:49 - hidden

- Status changed from To Be Closed to Closed

#24 - 2020-08-20 08:08 - hidden

- Description updated