Public Support - Support Request #11741

How to debug qml file

2020-07-17 10:33 - hidden

Status:	Closed		
Priority:	Normal		
Category:			
Customer:	AUDI	**Product Issue Numbers:**	
Department:	AST	**Affected Products:**	ADTF 3.7.0, ADTF Device Toolbox 3.1.0, ADTF Display Toolbox 3.4.0
Requester's Priority:	Normal	**Platform:**	Windows 10 64bit
Support Level:	2nd Level	**Topic:**	ADTF::Common
Resolution:	Solved Issue	**FAQ Links:**	

Description

Supportanfrage

In project we created a filter and we want to customize the option Create Pins from dat files to create pins from description file. I have used the sample qml(adtf_playback_input_filtereditor.qml) file from ADTF Player. But i am not able to open this in debug mode.

Lösung

It is not possible to debug qml files within Visual Studio.
There are some notes within the qml documentation on how to debug qml applications.
https://doc.qt.io/qt-5/qtquick-debugging.html

History

#1 - 2020-07-20 10:34 - hidden

- Project changed from Public Support to 11

- Status changed from New to Customer Feedback Required

- Topic set to ADTF::Common

- Customer set to AUDI

- Department set to AST

Hello,

What exactly are you open into Debug mode?
Are you trying to open adtf_playback_input_filtereditor.qml, or your own qml based on that

Do you get any error messages?

#2 - 2020-07-20 10:42 - hidden

Hello,
I am trying to open own qml developed based on adtf_playback_input_filtereditor.qml. I would like to debug it in Visual Studio. I added ADTF process to Visual Studi and opened my qml file and trying to add break points there. Its not showing any error but also not reaching breakpoints.

#3 - 2020-07-20 13:41 - hidden

- Status changed from Customer Feedback Required to In Progress

#5 - 2020-07-20 13:54 - hidden

- Status changed from In Progress to Customer Feedback Required

Hello Ganesh,

it is not possible to debug qml files within Visual Studio.
There are some notes within the qml documentation on how to debug qml applications.
https://doc.qt.io/qt-5/qtquick-debugging.html

Regards
Sebastian Stern

#6 - 2020-07-20 14:04 - hidden

Hello Sebastian,

Thank you.
Is there any sample qml files available to Create Pins from '.description' (by reading stream names) instead 'adtfdat'. Unfortunately i can find only adtf_playback_input_filtereditor.qml in adtf 3 installation folder. If no, ticket can be closed.

Regards,
Ganesh

#7 - 2020-07-20 14:07 - hidden

Hello Ganesh,

you can find all available functions within the documentation of ADTF.
https://support.digitalwerk.net/adtf/v3/adtf_html/class_editor_plugin_1_1_editor_plugin_base.html

Regards
Sebastian Stern

#9 - 2020-07-20 16:04 - hidden

Hi Ganesh,

as Sebastian already mentioned, there is no out of the box solution for this, all you have to know are the documented functions linked in the comment before.
What you have to is read in a .description file and create a a pin.
I dont think you need to debug or anything else, thats straight forward.

I would just ask why you decide this way ?
I would suggest following solutions, depending on you use case (maybe you should give us more details about that):

- Fixed structures:
  - Generate a header from description file and create pins for each structure (https://support.digitalwerk.net/adtf/v3/adtf_html/mainpage_mediadescription_pkg.html)
    - PRO: Defined scope, valid data/structure guaranteed, accessing whats available
    - CON: Not generic, build required
- Scripting Filter (https://support.digitalwerk.net/adtf/v3/adtf_html/page_javascript_filter.html)
  - Use JavaScript and create pins from description files
    - PRO: Can be done in ADTF Configuration Editor, generic, adaptable, no rebuild
    - CON: No closed binary, Javascript instead of C++, hard to debug/analyze

#10 - 2020-07-24 11:55 - hidden

Hallo Florian,

in unserem UseCase möchten wir mittels des Kontextmenüs in Configuration-Editor Filterpins erstellen. Welche Filterpins erstellt werden sollen, ist in einem Description-File spezifiziert, der in den Filterproperties angegeben ist.
Kannst du uns sagen, wie und wo sich die Logausgaben vom QML-File anzeigen lassen, da sich der QML-File nicht direkt debuggen lässt?
Hier ist unser Beispiel QML-Code:

```
import QtQuick 2.7
import QtQuick.Controls 1.4
import QtQuick.Layouts 1.1
import QtGraphicalEffects 1.0
import QtQuick.Dialogs 1.2

import Dialog.Input 1.0
import Dialog 1.0
import EditorPlugin 1.0

import Utils 1.0
import ModelTree 1.0
import Helpers 1.0

EditorPluginBase
{
    onExecute:
    {
       console.log("Start")
       createOutputPin("MyOutPin");
```

```
        createInputPin("MyInPin");
        console.log("End")
    }

}
```

Lassen sich aus dem QML-File auch direkt eigene Methoden des Filterplugins aufrufen bzw zum Aufruf durch QML registrieren, da wir das Parsen des Description-Files schon in C++ Code implementiert haben.

Danke.

Viele Grüße

Dirk

#11 - 2020-07-27 20:50 - hidden

Hallo Dirk,

im Moment ist unter Windows die einzige Möglichkeit JavaScript Log Ausgaben zu lesen,
den Configuration-Editor mit dem Parameter "-v" zu starten und dann mit VisualStudio und "Attach to process" die Log-Ausgaben auf der VisualStudio-Output-Konsole zu verfolgen.
Wir haben es auch schon auf dem Schirm, dass das etwas umständlich ist und überlegen intern schon wie wir das Debuggen hier verbessern können.

Es ist leider nicht Möglich aus der QML-Ebene des Configuration-Editors Filter-Code aus deinem *.adtfplugin auszuführen.
Der Configuration-Editor kann jedoch mit eigenen Modulen sog. *.adtfceplugin Dateien erweitert werden.

Dazu findest du in den Guides ein Beispiel:
https://support.digitalwerk.net/adtf/v3/guides/tutorial_ce_module.html

Anstatt eines neuen Property-Editors könnte man zum Beispiel ein QML-Singleton registrieren, der einige Helper-Funktionen bereitstellt.
Diese stehen dann auch innerhalb der Ausführung eines Filter-Editors zur Verfügung.

Viele Grüße
Sebastian

#13 - 2020-08-05 15:30 - hidden

Hallo Sebastian,

ich habe ein Frage zu der Verknüpfung QML mit einer C++ Lib. Ich habe es jetzt mal so wie in den QT Beispielen umgesetzt und das funktioniert auch, d.h. ich habe mir ein eigenes QQmlExtensionPlugin geschrieben auf das ich per QML-File aus dem Kontext-Menü eines Filters zugreifen kann (wenn ich den QML-File in der Plugindescription des Filters angebe). Das QQmlExtensionPlugin mache ich dem QML-File über die Datei QMLdir und den Befehl plugin XY bekannt. Dafür muss ich das QQmlExtensionPlugin und die QMLdir Datei aber innerhalb des ADTF-Ordners ablegen.
Gibt es auch eine Möglichkeit in ADTF die Suchpfade nach den QQmlExtensionPlugs zu setzen? In meinem Standalone QT-Testprojekt konnte ich dies über die QML Engine und den Befehl addImportPath setzen.
Die einzige Möglichkeit, die ich bis jetzt gefunden habe, ist eine Umgebungsvariable QML2_IMPORT_PATH anzulegen.

Wenn ich es stattdessen über das ADTF Beispiel "demo_property_editor" umsetze würde, habe ich noch nicht ganz verstanden wie es damit funktioniert. So wie ich es verstehe werden durch den File module.qrc alle qml-Files in das *.ceplugin gelinkt.
Aber ich verstehe nicht, warum in der Datei qmldir dann ein QML-File StartMain.qml aufgerufen wird (da diese Datei nicht existiert) oder ist dieser File von euch in den ADTF Configuration Editor gelinkt worden?
Muss nur das *.ceplugin in den ADTF Suchpfad kopiert werden damit ADTF bzw QML darauf zugreifen kann oder noch andere Dateien, wie der QML-File oder die Datei qmldir?
Falls ich keine QML-Files in den ADTF-Ordner kopieren muss (da sie schon in die dll gelinkt worden sind), wie lässt sich dann die Funktionalität des *.ceplugins aus dem Kontextmenü eines Files aufrufen (so wie ich es wie oben beschrieben mit dem QML-File und dem QQmlExtensionPlugin umgesetzt habe)? Sprich was schreibe ich dann in die Plugindescription des Filters?

Viele Grüße

Dirk

#14 - 2020-08-05 21:54 - hidden

Hallo Dirk,

innerhalb des Configuration-Editors kannst du in den Optionen(Alt+O) unter <Modules> Suchpfade eintragen.
In diesen Pfaden sucht der Configuration-Editor beim Start nach *.adtfceplugin Dateien.
Alle Gefundenen werden geladen und eine Main.qml darin gesucht.

Diese Main.qml Dateien werden dann als Einstiegspunkt ausgeführt Dort kannst du zum Beispiel dann Helper instanziieren.
Wenn du deine Qml-Klasse/Datei zu Beginn mit dem dem Keyword <pragma Singleton> versiehst, dann ist diese Klasse global verfügbar ohne Instanz.
Man kann auch C++ Klassen als QML-Singleton mit in ein Plugin packen.

Die Umgebungsvariable und die Qt Funktionen wie <addImportPath> oder <addPluginPath> musst du gar nicht benutzen.
Das Beispiel ist im Prinzip auch nur ein QQmlExtensionPlugin mit einer qrc und alles wird dann in einer Lib verpackt mit der Dateiendung *.adtfceplugin.

Ich hoffe ich konnte dir etwas helfen.
Solltest du weitere Fragen haben, dann melde dich einfach.

Viele Grüße
Sebastian

#15 - 2020-08-11 13:42 - hidden

Hallo Dirk,

sollte es hier keine offenen Punkte mehr geben würde ich das Ticket demnächst schließen.
Sollte doch noch etwas unklar sein, dann einfach ein neues Ticket anlegen.

Vielen Dank & Grüße
Sebastian Stern

#16 - 2020-08-17 09:01 - hidden

- Status changed from Customer Feedback Required to To Be Closed

- Resolution set to Solved Issue

#17 - 2020-08-18 10:20 - hidden

- Description updated

#18 - 2020-08-18 10:21 - hidden

- Project changed from 11 to Public Support

- Private changed from Yes to No

#19 - 2020-10-06 10:27 - hidden

- Status changed from To Be Closed to Closed