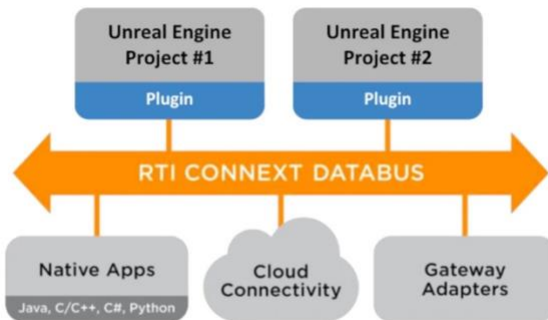


Simplified Real Time Data Sharing

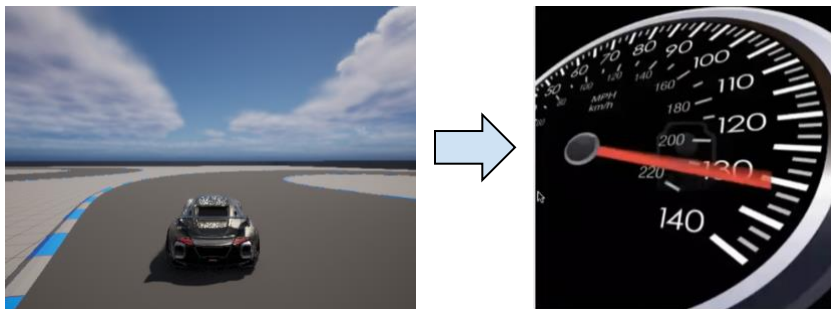
Unreal Engine plugin for RTI Connex

Introduction

This plugin makes it easy to integrate your Unreal Engine project with other applications that you have developed using RTI Connex.



The plugin is ideal for those projects that need to publish information in real time that is received and processed by other applications. The data types are defined in XML and a code generator automatically creates the C++ source code so that it can be incorporated into the Unreal Engine project. All actions are implemented in the Blueprint without the need to code anything in C++.



This tutorial allows you to experiment with the plugin by following the instructions which will take approximately 1 hour¹. It is not necessary to have previous experience with Unreal Engine or RTI Connex, since it explains how to create a project from scratch and includes explanatory screenshots of the entire process.

¹Please note that the software listed in the prerequisites is very large and could take a couple of hours to install if you do not already have it on your computer.

Prerequisites

Below is the list of software and hardware prerequisites that you will need during this tutorial:

- Unreal Engine 5.1.1 or 5.2.0
- RTI Connex Professional 6.1.1 or 6.1.2; or RTI Connex Drive 2.0.1
- Microsoft Visual Studio 2022 (Community version is fine)
- A Windows² PC powerful enough to run Unreal Engine
 - System requirements are listed at: <https://docs.unrealengine.com/5.1/en-US/hardware-and-software-specifications-for-unreal-engine/>
- 70 GB of free disk
 - RTI Connex ~2 GB
 - Unreal Engine (including the project that we will create) ~60 GB
 - Microsoft Visual Studio Community ~5.5 GB

²Similar tutorials for Linux and Mac are currently under preparation.

Installing the prerequisites

Unreal Engine

Install Unreal Engine 5.1.1 or 5.2.0. The software is available at no cost:

<https://www.unrealengine.com>

Unreal Engine is installed using the Epic Games Launcher, which can be downloaded from the unrealengine.com website:



1. Double click on `EpicInstaller-14.6.2-unrealEngine.msi` and follow the instructions to install the Launcher software on your PC. You may need to create a login account with Epic Games to access the content in the Launcher.
2. Open the Epic Games launcher, select 'Unreal Engine' in the sidebar, and click on the **Library** tab.
3. Click on the '+' sign to add an engine, select version 5.1.1 or 5.2.0 and click on **Install**. Note that the link will change to version 5.1.1 or 5.2.0. An end-user license agreement will appear, which must be agreed to begin the download and installation.

Be aware that depending on the computer characteristics, the installation can take over an hour.

RTI Connex

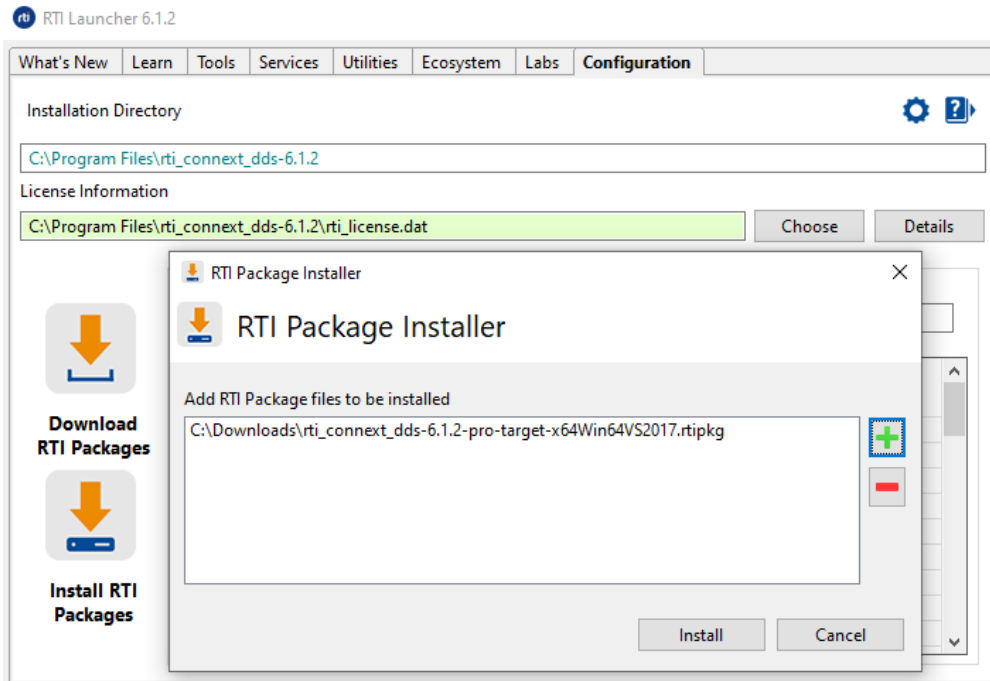
Install RTI Connex Professional 6.1.1 or 6.1.2 or RTI Connex Drive 2.0.1 on your computer. If you need a trial version, you can request it at the following link:

<https://bit.ly/43dSUfo>

Install and configure RTI Connex using the following steps:

4. Run the `rti_connex_dds-6.1.1-pro-host-x64Win64.exe` installer and follow the onscreen instructions to install the software on your PC. If using the **Free Trial** version of Connex 6.1, the name of the installer is `rti_connex_dds-6.1.1-lm-x64Win64.exe` instead.
5. When installation finishes, there is an option to 'Open the RTI Launcher'. Ensure this option is enabled – the RTI Launcher will then open.
6. Click on the **Configuration** tab in the RTI Launcher.
7. If the 'License Information' field is empty with a red background, click the 'Choose' button to open an explorer and locate the `rti_license.dat` file that was provided to you by RTI.
8. If using the **Free Trial** version of Connex 6.1, no further steps are required.
9. Click on **Install RTI Packages**, then the '+' sign to browse your local disk to locate the file: `rti_connex_dds-6.1.1-pro-target-x64Win64VS2017.rtipkg`. This file contains the DDS target libraries and is typically paired with the host installer of Step 1. Press 'Install'.





Microsoft Visual Studio

Install Microsoft Visual Studio 2022. The Community version is available at no cost:

<https://visualstudio.microsoft.com>

1. Download and run the installer for Visual Studio 2022: `visualstudiosetup.exe` and follow the instructions to install the software on your PC
2. Make sure that you select the following packages during the installation:
 - Desktop development with C++ (leave the default options)
 - Individual components:
 - .NET Framework 4.8.1 targeting pack
 - .NET Framework 4.8.1 SDK

Environment variables

You will need to add in the System environment variables the full path to the RTI Connex software:

1. Click on the Windows Start menu and type the word **variables** so that it shows **Edit the system environment variables** and click on it.
2. Click on **Environment Variables**
3. In the System variables section, create a new variable `NDDSHOME` with value `C:\Program Files\rti_connex_t_dds-6.1.1` **or** `C:\Program Files\rti_connex_t_dds-6.1.2`

Note: make sure that the path really matches the location where you installed the software.

You will also need to add in the System `PATH` the full path to the DotNet and Python tools:

1. In the System variables section, look for the variable `PATH` and click on **Edit**
2. Click on **New** and add the following entry:
`C:\Program Files\Epic Games\UE_5.1\Engine\Binaries\ThirdParty\DotNet\6.0.302\windows`
3. Click on **New** once again and add the following entry:
`C:\Program Files\Epic Games\UE_5.1\Engine\Binaries\ThirdParty\Python3\Win64`

Note: if your system already has an installation of Python, be sure to move this item to appear in the `PATH` before any other installation of Python. This can be done with the **Move Up** button in the path editing window.

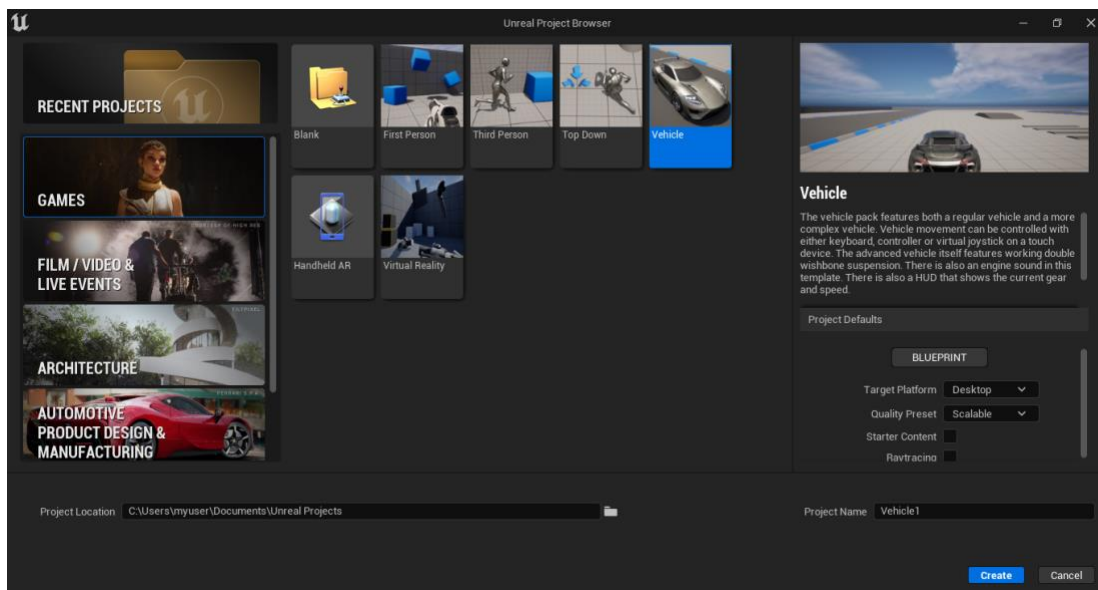
Make sure that the paths really match the location where you installed the software. In order to test that both tools work correctly:

1. Click on the Windows Start menu and type the word **command** so that it shows **Command Prompt** and click on it.
2. In the command prompt, type `dotnet` and press enter. It should display a brief help on the usage of the tool.
3. In the command prompt, type `python --version` and press enter. It should display the version of the tool, which was 3.9.7 as of this writing.
4. Keep this window open, we will need it later.

Creating the Unreal Engine project

In order to create a functional project in the shortest time possible, we will take advantage of the nice Vehicle template that comes with Unreal Engine 5:

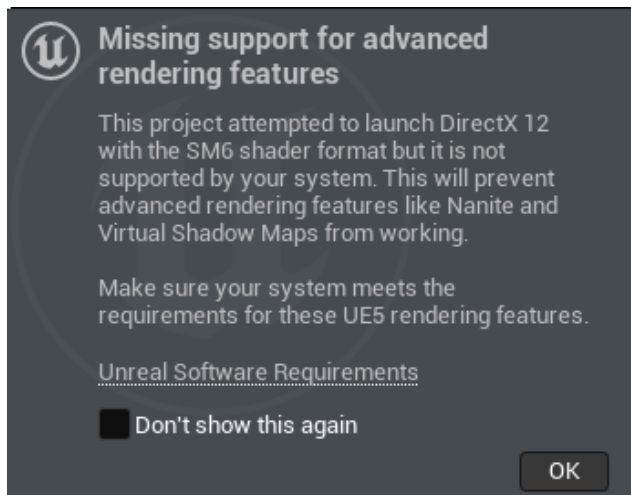
1. Open the Epic Games launcher.
2. Click on the **Launch Unreal Engine 5.1.1 or 5.2.0** button (in the upper right corner of the launcher). This will open the **Unreal Editor** and close the Launcher. Note that when the editor is launched, there may be some setup delay due to compiling shaders, etc.
3. Click on **GAMES** and select **Vehicle**.
4. Leave the Project Location with the default value. Fill in the required details on the right-hand side as:
 - Target Platform: Desktop
 - Quality Preset: Scalable
 - Starter Content: unchecked
 - Raytracing: unchecked
 - Project Name: Vehicle1



Click on **Create** to continue.

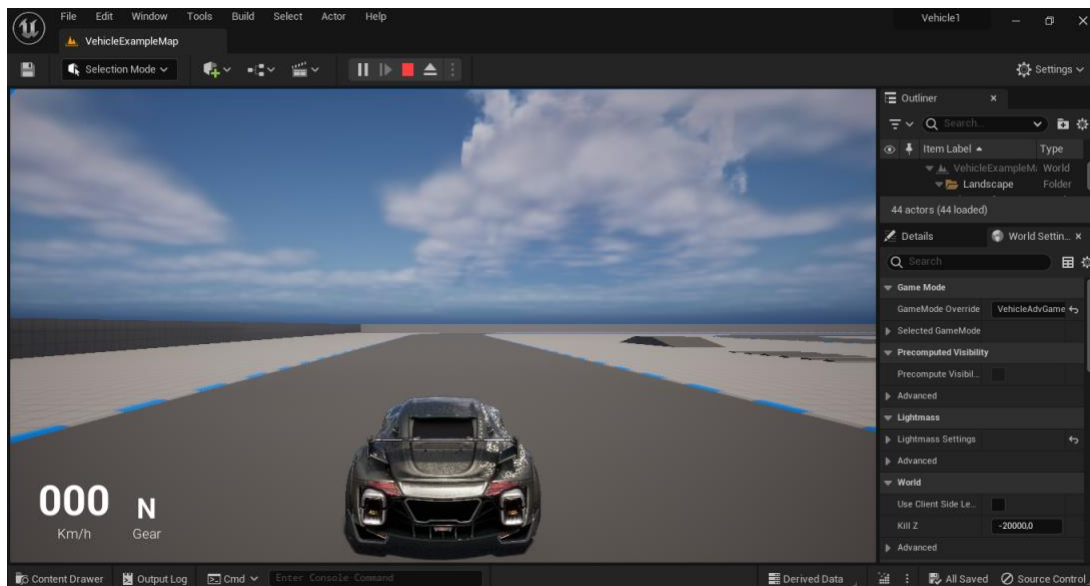
5. The project automatically opens in the Unreal Editor. Note that if certain software/hardware dependencies are not found, they may be flagged when the editor

opens using a dialog box, such as:



Note also that some of these warnings are about the performance capabilities of your system, but the applications will still work correctly.

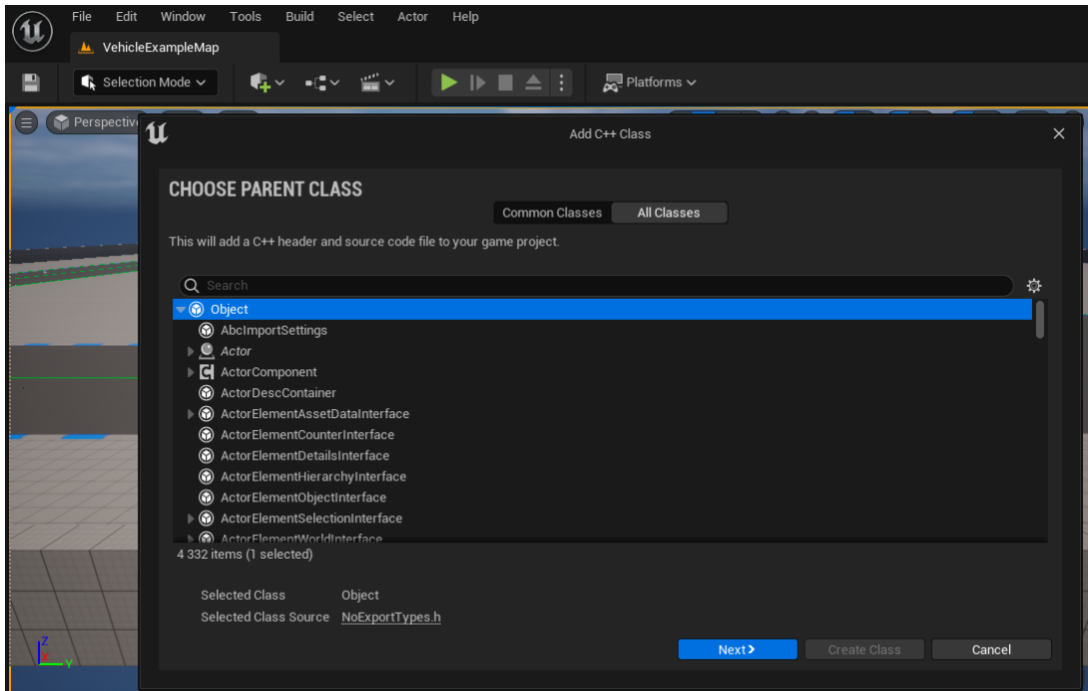
6. To check that everything works as expected, click on the green **Play** button on the upper side, then click once in the middle on the screen, and finally try to drive the car using the arrow keys. Use the **Escape** key to finish your test drive.



The project is created as a Blueprint, but we need to convert it into a C++ project because we are going to add C++ code later.

To force the conversion into a C++ project:

1. Click on **Tools > New C++ Class**
2. Click on the **All Classes** tab.
3. Select **Object** and click on **Next**.
4. Leave the name of the class with the default value and click on **Create Class**



5. At this point the project has been converted. It displays the following message:

Project now includes sources, please close the editor and build from your IDE.

Click on **OK**, but do not close the Unreal Editor yet!

6. Next it displays the following message:

Successfully added class 'MyObject', however you must recompile the 'Vehicle1' module before it will appear in the Content Browser. Would you like to edit the code now?

Click on **No**. Again, do not close the Unreal Editor yet since we are going to use it just below

Installing the Unreal Engine plugin for RTI Connex

There are two different methods to install the plugin. If possible, use the first method to download it and install it from the marketplace. Use the second method to install it manually only if you need to test a custom version that has been delivered to you.

Installation from the Unreal Engine marketplace

1. Click on **Edit > Plugins**
2. In the search box, type the word `simplified` so that it shows **Simplified Real Time Data Sharing** and click on it
3. It displays a message asking for confirmation to enable the plugin. Click on **Yes**
4. At this point, the plugin has been installed in
`C:\Users\myuser\Documents\Unreal Projects\Vehicle1\Plugins\ConnexPlugin`
5. Close the Unreal Editor

Manual installation

1. Close the Unreal Editor before using this method.
2. Open a file explorer and go to `C:\Users\myuser\Documents\Unreal Projects\Vehicle1`
3. Create a folder named `Plugins`
4. Copy the `Simplified_Real_Time_Data_Sharing_v1.1.zip` file that has been delivered to you into folder `C:\Users\myuser\Documents\Unreal Projects\Vehicle1\Plugins`
5. Unzip the file. It will automatically create the folder `Simplified_Real_Time_Data_Sharing_v1.1`
6. Delete the zip file and rename the folder to `ConnexPlugin`

Name	Date modified	Type
Config	27/03/2023 17:45	File folder
Docs	27/03/2023 19:36	File folder
Resources	28/03/2023 8:12	File folder
Source	14/02/2023 17:55	File folder
Tools	28/03/2023 19:30	File folder
ConnexPlugin.uplugin	28/03/2023 19:21	UPLUGIN File
README.md	28/03/2023 19:29	MD File

Generating the C++ source code

You can define the data types in XML, then use a code generator to automatically create the C++ source code that you need to incorporate in the Unreal Engine project.

In this tutorial, we are using the CarInfo.xml which contains a data type with three fields:

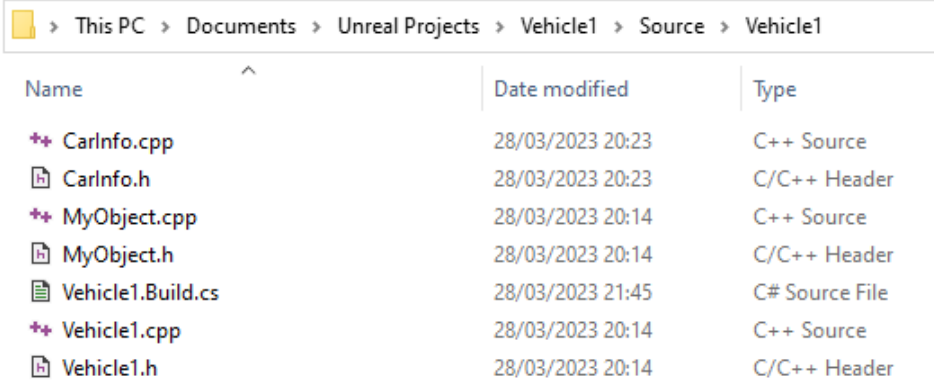
- `coordinates`: the 3D coordinates of the car in the Unreal Engine world
- `speed`: the speed of the car in mph
- `id`: the unique identifier of the car

```
<types>
<struct name="Position">
  <member name="x" type="float32"/>
  <member name="y" type="float32"/>
  <member name="z" type="float32"/>
</struct>
<struct name="CarInfoData">
  <member name="coordinates" type="nonBasic" nonBasicTypeName="Position"/>
  <member name="speed" type="float32"/>
  <member name="id" type="string" stringMaxLength="100" key="true"/>
</struct>
</types>
```

In order to generate the C++ source code, follow the steps below:

1. Using the command prompt that you had opened before, type `cd C:\Users\myuser\Documents\Unreal Projects\Vehicle1\Plugins\ConnexPlugin\Tools` and press enter.
2. Type `python generator_tags.py -xml ..\Config\CarInfo.xml -c CarInfo -p Vehicle1 -v 5` and press enter.

- xml: indicates the path of the XML file
 - c: class name that you want to use in the Unreal Engine project for that data type
 - p: indicates the Unreal Engine project name
 - v: indicates the Unreal Engine version, Unreal Engine 5 in this case
3. At this point, the `CarInfo.cpp` and `CarInfo.h` files have been generated. Copy them into folder `C:\Users\myuser\Documents\Unreal Projects\Vehicle1\Source\Vehicle1`

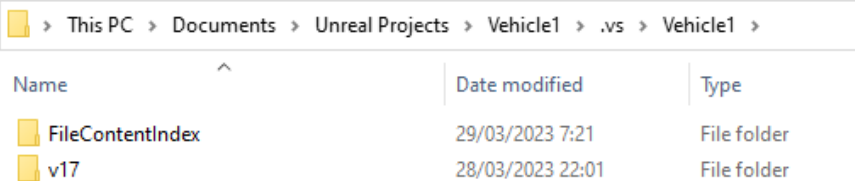


Name	Date modified	Type
CarInfo.cpp	28/03/2023 20:23	C++ Source
CarInfo.h	28/03/2023 20:23	C/C++ Header
MyObject.cpp	28/03/2023 20:14	C++ Source
MyObject.h	28/03/2023 20:14	C/C++ Header
Vehicle1.Build.cs	28/03/2023 21:45	C# Source File
Vehicle1.cpp	28/03/2023 20:14	C++ Source
Vehicle1.h	28/03/2023 20:14	C/C++ Header

4. In the same folder, edit the file `Vehicle1.Build.cs`. In line 11, add `"ConnexPlugin"` and `"RTIConnexLibrary"` right before `"Core"`:

```
PublicDependencyModuleNames.AddRange(new string[] {
"ConnexPlugin", "RTIConnexLibrary", "Core", ...
```

5. Using a file explorer, go to folder `C:\Users\myuser\Documents\Unreal Projects\Vehicle1`. Right click on the file `Vehicle1.uproject` and select **Generate Visual Studio project files**. It creates the `.vs` folder.



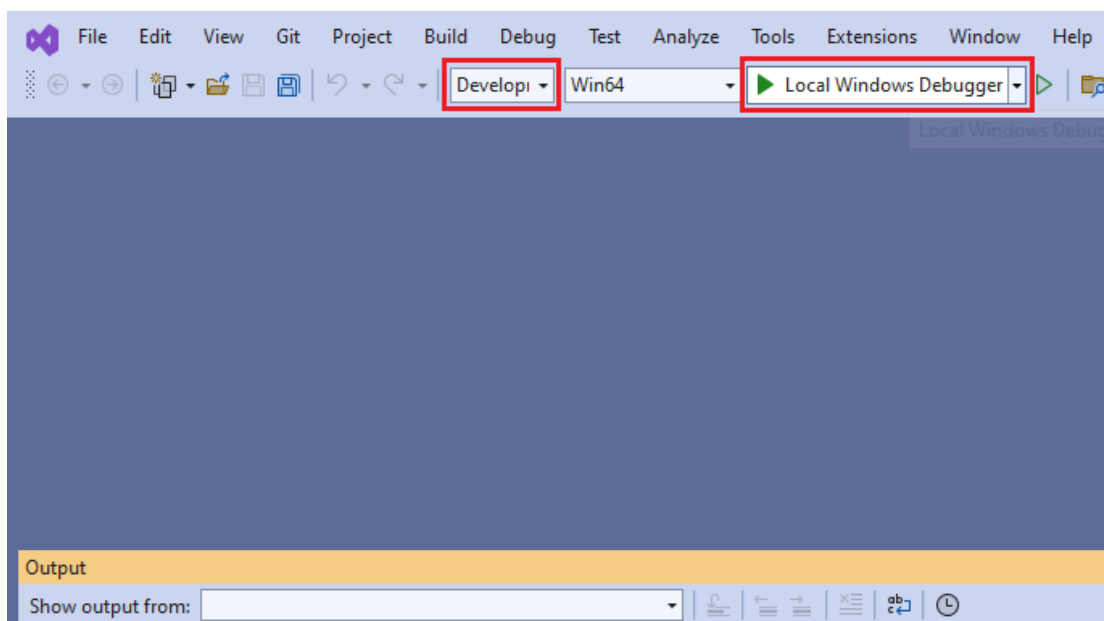
Name	Date modified	Type
FileContentIndex	29/03/2023 7:21	File folder
v17	28/03/2023 22:01	File folder

Compiling the Unreal Engine project

We already have everything necessary to compile the project:

1. Using a file explorer, go to folder `C:\Users\myuser\Documents\Unreal Projects\Vehicle1`
2. Double click on the `Vehicle1.sln` file to open the project in Visual Studio
3. In the dropdown boxes in the upper side, select **Development Editor** and **Local Windows Debugger**. Then click on the green **Play** button to launch the build.


Note: If you get the following error message: “Unable to instantiate module: 'ConnexPlugin': System.ArgumentNullException: Value cannot be null. (Parameter 'path1')” make sure that the path in the `NDDSHOME` environment variable is correct (see section `Environment variables` above).

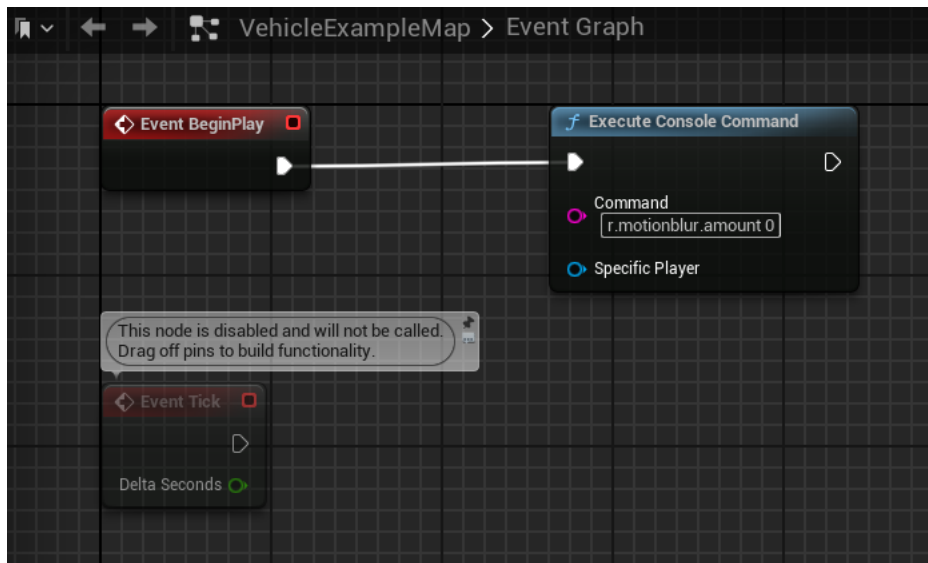


When you are compiling for the first time, it may take a while. When it is finished, it will automatically open the project in the Unreal Editor.

In case that this does not happen, look for any compilation errors in Visual Studio’s output box.

Publishing real-time information on the databus

1. Click on the Blueprint icon  on the upper toolbar and select **Open Level Blueprint**
2. It shows an example which includes two of the typical events that every developer will need to implement:
 - Event BeginPlay: it executes just once at startup.
 - Event Tick: it executes periodically and you can specify how often

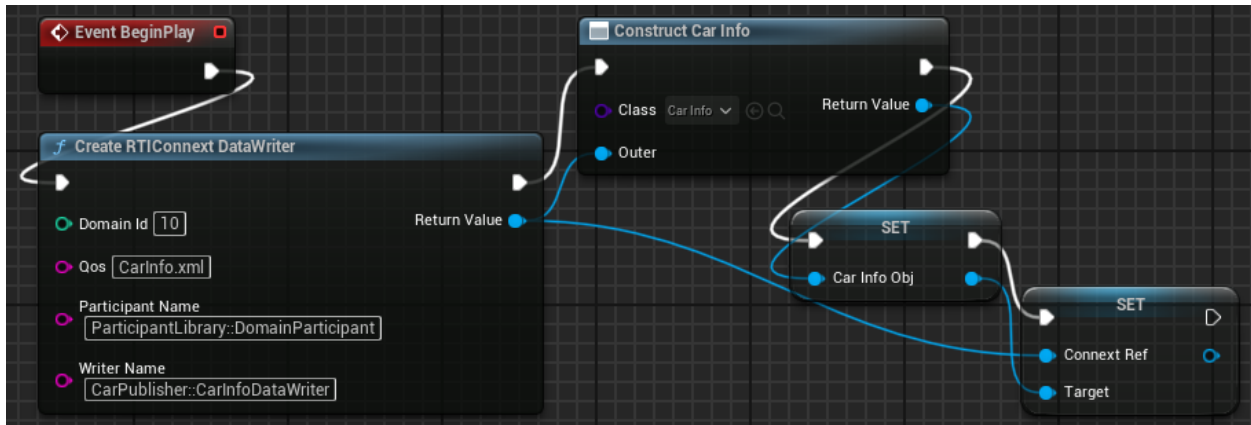


3. Right click on the **Execute Console Command** node and delete it, since we will not need it.

You can adjust the zoom level using your mouse wheel, and you can drag the work area while keeping your mouse right button pressed.

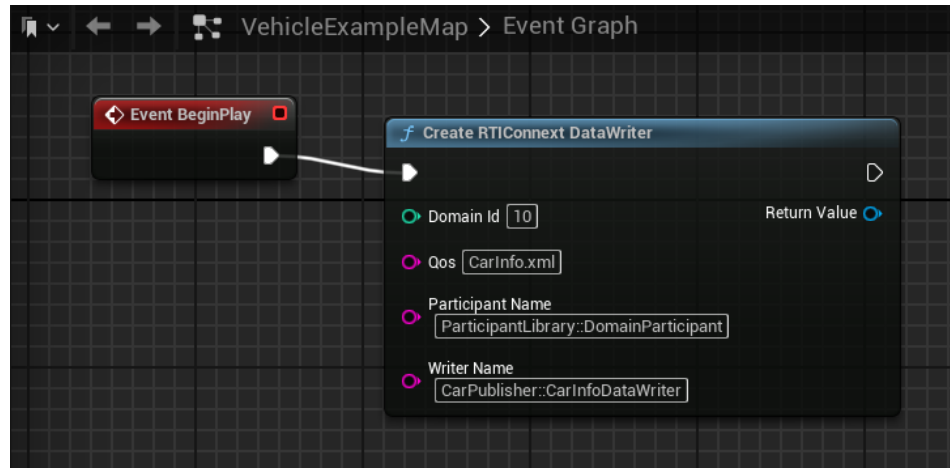
Event BeginPlay

This event requires 4 nodes that we are going to create next.

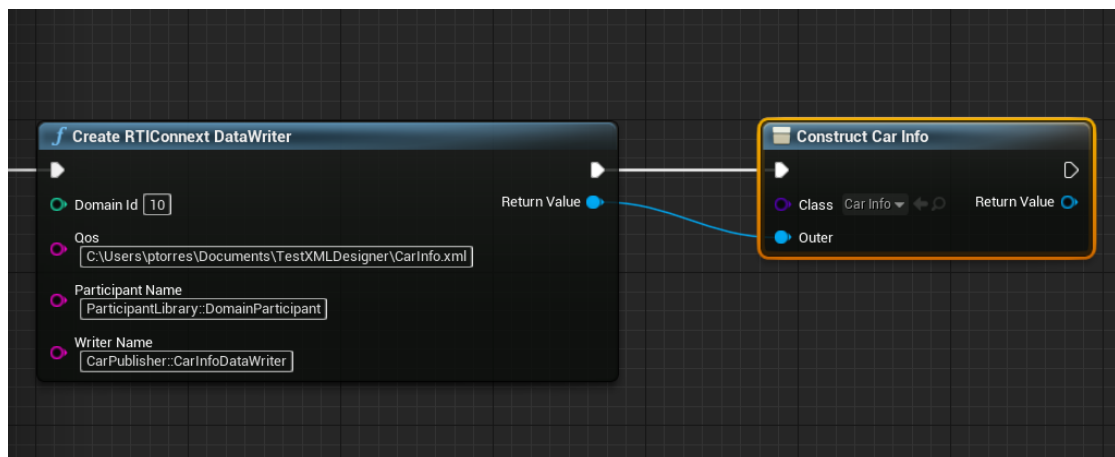


Follow the steps below:

1. Right click in an empty space in the work area and type `Create RTIConnex DataWriter` in the search box until it shows the element that we need; click on it to create a new node. This function is in charge of creating the DDS DataWriter entity together with the DDS participant if it is not already created. The information used by the function comes from both the parameters and the DDS xml.
2. Fill in the required details:
 - **Domain Id:** indicate the domain where all your RTI Connex applications will be running so that they can communicate with each other; type `10`
 - **Qos:** file which contains the quality of service parameters for the communication, and the entities, topic, types definition; they are in the same file where we had defined the data types above, type `CarInfo.xml`
 - **Participant Name:** Name of the participant to use with its corresponding library; type `ParticipantLibrary::DomainParticipant`
 - **Writer Name:** Name of the DataWriter with its corresponding Publisher from the xml, which writes `CarInfo` data type that we had defined above; type `CarPublisher::CarInfoDataWriter`
3. Connect the **Event BeginPlay** node with the **Create RTIConnex DataWriter** node.

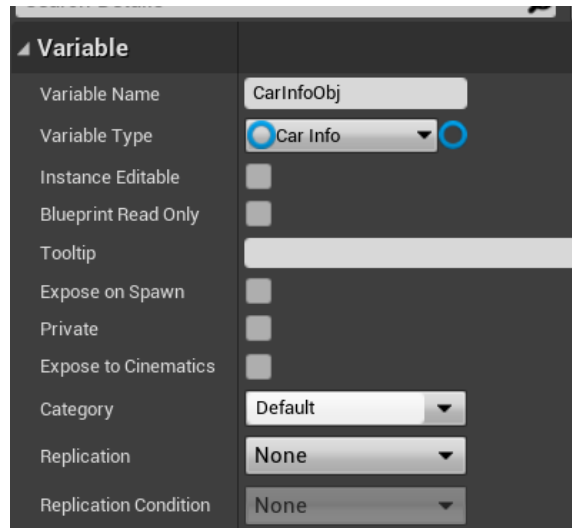


4. Right click in the work area and type `Construct object from class` in the search box; click on it to create a new node.
5. Click in the dropdown and search for the class created early.
 - o Class: `CarInfo`
6. Connect the **Create RTIConnex DataWriter** node with the **Construct Car Info** node. Also connect the **Return Value** pin with the **Outer** pin.

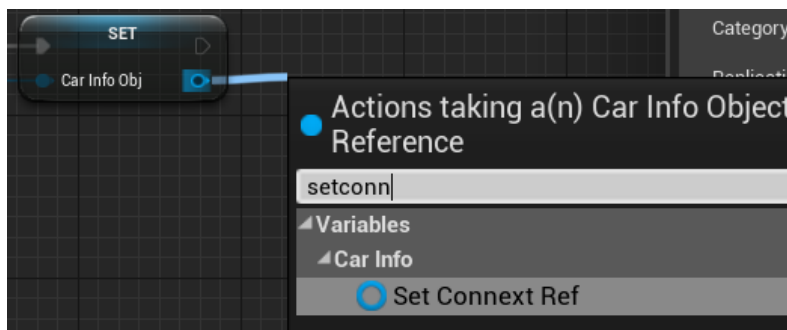


7. This object initialization/creation is necessary because the plugin does not know beforehand the objects that we have imported. Once the object is instantiated you can use its functions.

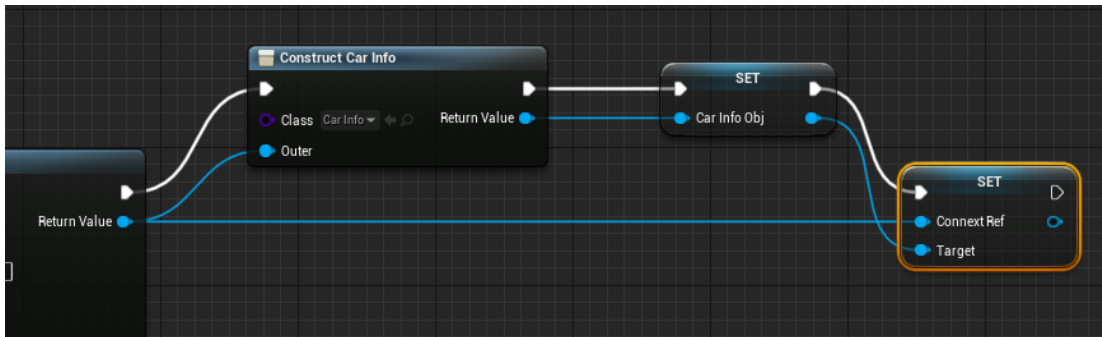
8. Promote the new object to a variable so it is easier to use inside the Blueprint. In the **Construct Car Info** node, right click on **Return Value** and select **Promote to Variable**.
9. It opens a Details window on the right hand side. Fill in the required details:
 - Variable Name: CarInfoObj
 - Variable Type: Car Info



10. Draw a line starting from the output of the node that we just created and ending in an empty space in the work area. Type `SetConnnextRef` in the search box; click on it to create a new node.

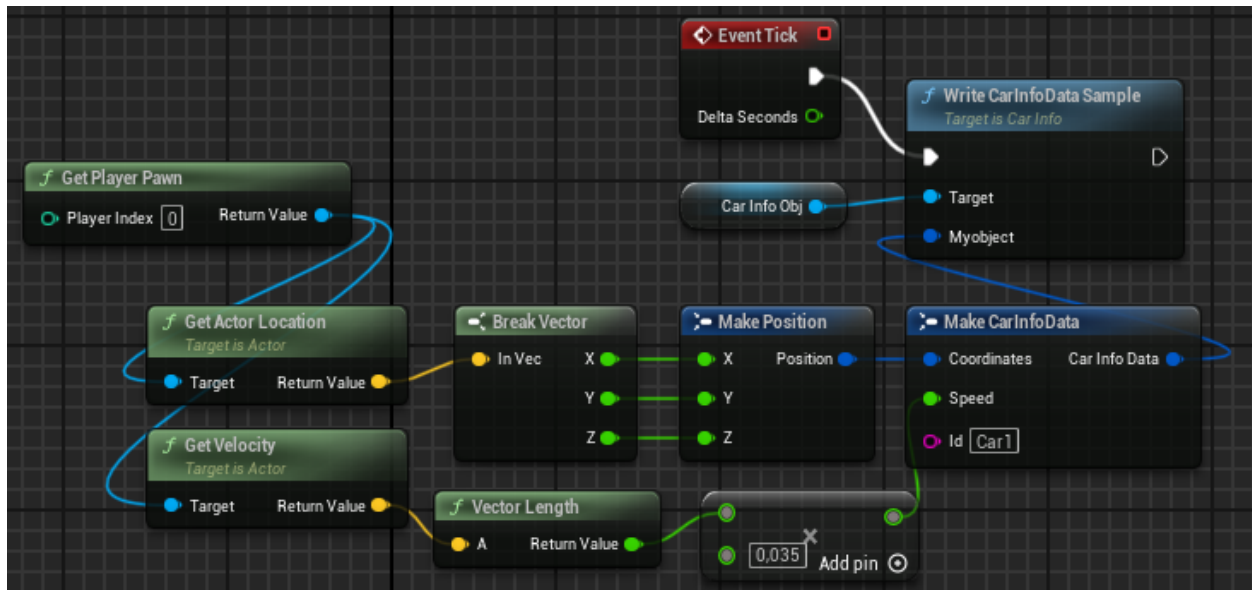


11. Connect the last two nodes that we have created. Also connect the **Return Value** pin (of the **Create RTIConnnext DataWriter** element) with the **ConnnextRef** pin.



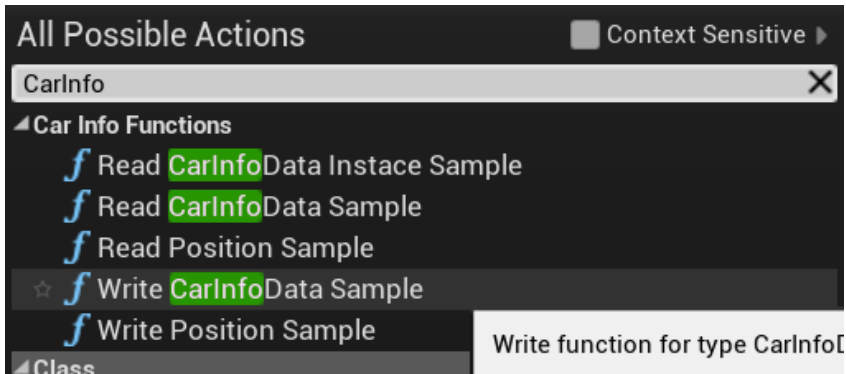
Event Tick

This event requires 10 nodes that we are going to create next.

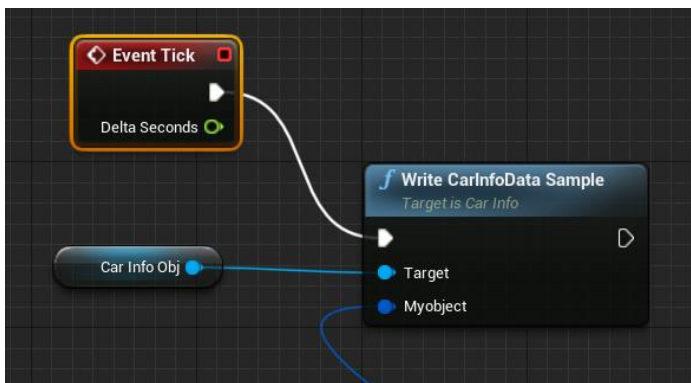


Follow the steps below:

1. Right click in the work area to open the search box, and ensure that the **Context Sensitive** option is unchecked. Now type `Write CarInfoData Sample` in the search box. Click on it to create a new node. This unreal function is in charge of writing the data to the dds databus using the type defined in the xml file, and the writer declared in the Event BeginPlay.



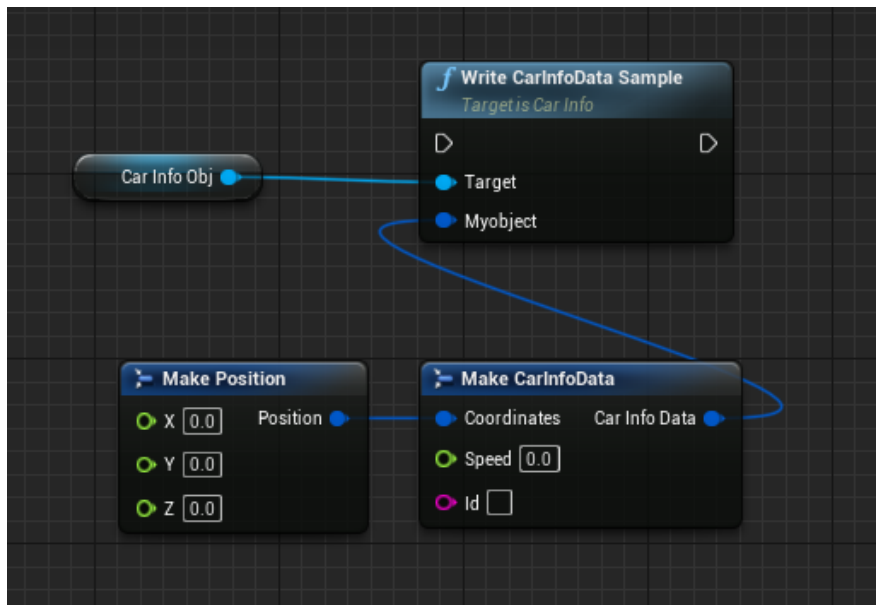
2. Right click in the work area and type `Get CarInfo Obj` in the search box; click on it to create a new node.
3. Connect the **CarInfo Obj** node with the **Target** pin. Also connect the **Event Tick** node with the **Write CarInfoData Sample** node.



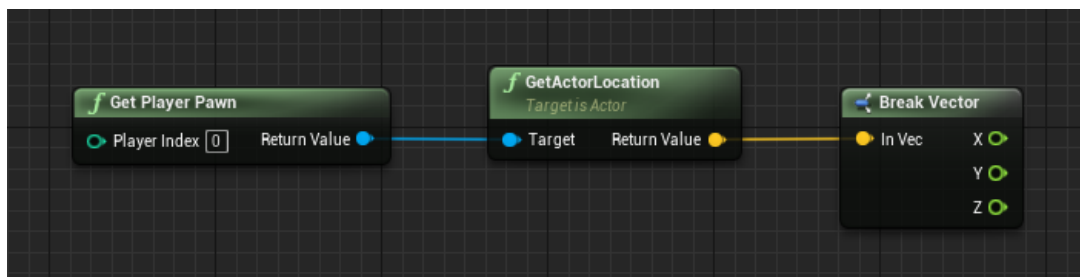
4. Next, we will create the Unreal struct (UStruct) equivalents of the two data types that we had in the XML file: `Position` and `CarInfoData`. We will use them to write data to the Connex DDS databus.

Right click in the work area and type `Make Position` in the search box; click on it to create the element.
5. Right click in the work area and type `Make CarInfoData` in the search box; click on it to create the element.
6. Fill in the required `CarInfoData` details:
 - o Id: `Car1`

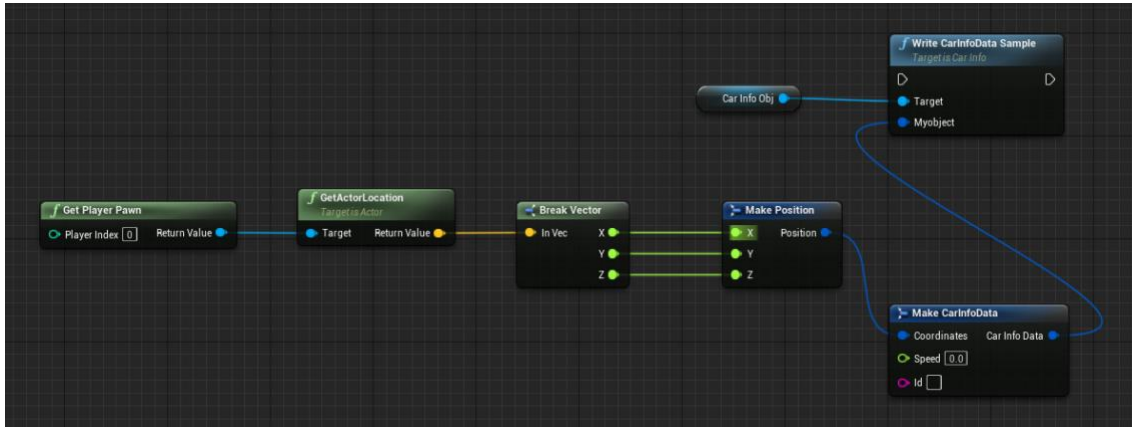
7. Connect the **Position** pin with the **Coordinates** pin. Also connect the **MyObject** pin with the **CarInfoData** pin.



8. Right click in the work area and type `Get Player Pawn` in the search box; click on it to create the element.
9. Right click in the work area and type `Get Actor Location` in the search box; click on it to create the element.
10. Right click in the work area and type `Break Vector` in the search box; click on it to create the element.
11. Connect the **Return Value** pin (of the **Get Player Pawn** node) with the **Target** pin. Also connect the **Return Value** pin (of the **Get Actor Location** node) with the **In Vec** pin.



12. Next, to join the nodes that we had created before with these last ones, connect the X, Y and Z pins.



13. Right click in the work area and type `Get Velocity` in the search box; click on it to create the element.
14. Right click in the work area and type `Vector Length` in the search box; click on it to create the element.
15. Right click in the work area and type `Multiply` in the search box; click on it to create the element.

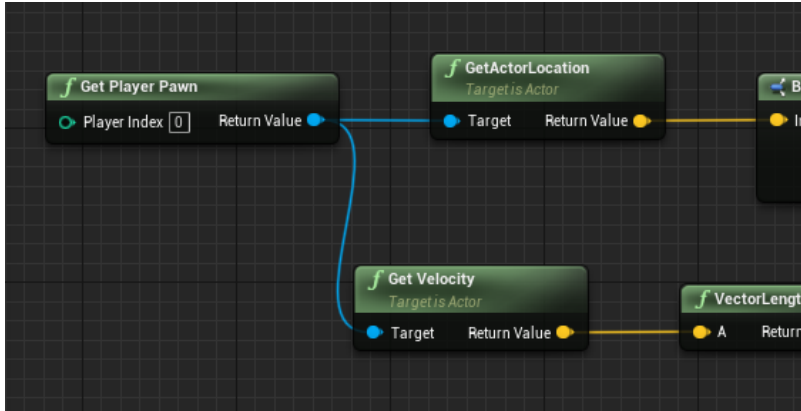
16. Make the following Connections:

- The **Return Value** pin (of the **Get Velocity** node) with the **A** pin.
- The **Return Value** pin (of the **Vector Length** node) with the **Multiply** node.
- The **Multiply** node with the **Speed** pin.

17. Edit the second multiplication factor to be: 0.035 (conversion from cm/s to Km/h)
The resulting diagram should look like this:



18. Connect the **Get Player Pawn** node with the **Get Velocity** node.



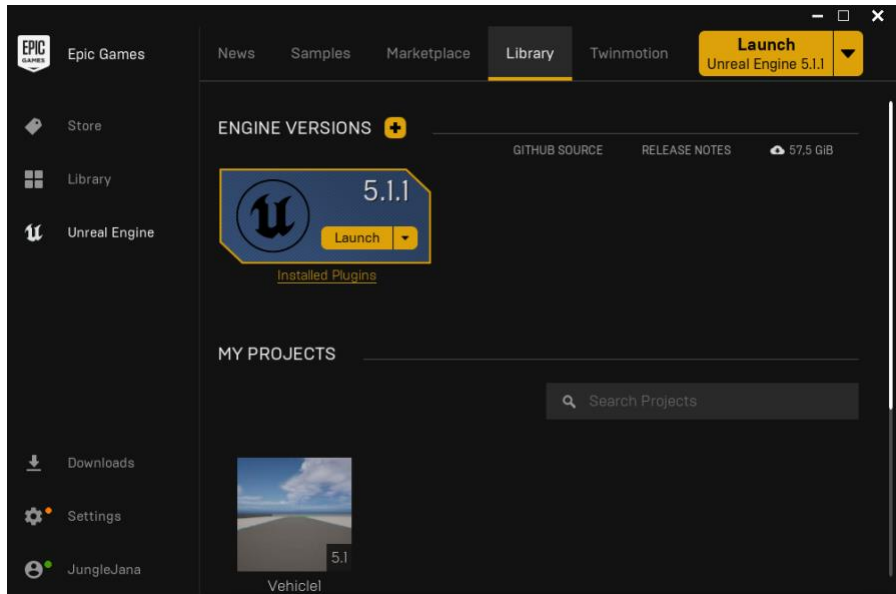
19. Click on **File > Compile**

20. Click on **File > Save All** and close the Blueprint editor

21. Back to the Unreal Editor, click on the **Play** button to run the project. This VehicleExample should now be exporting position and speed using RTI Connex!

22. Click on **Output Log** at the bottom of the screen and take a quick look at the last few lines to make sure that there were no errors during the startup

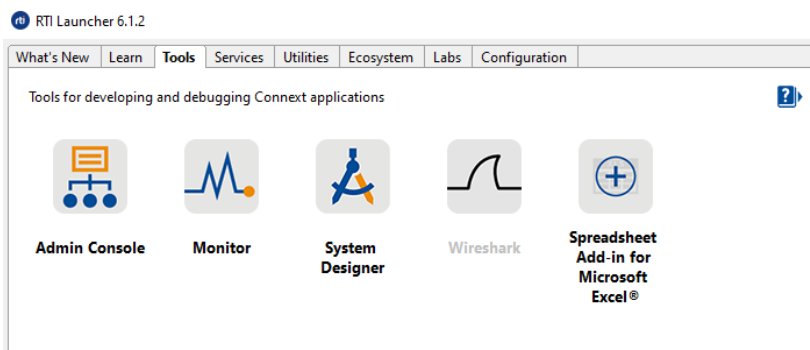
For future sessions you will no longer need to start the project from Visual Studio. You can do this directly from the **Library** tab of the Epic Games launcher, simply by double-clicking on Vehicle1.



Subscribing to receive real-time information from the databus

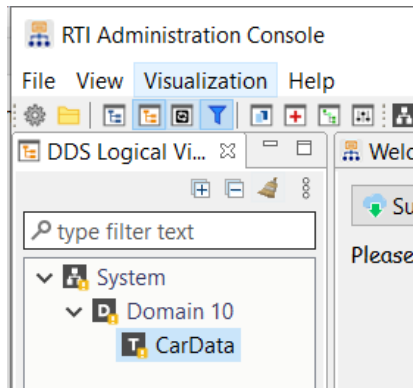
The VehicleExample is now publishing its speed and position data to a DDS databus using RTI Connex. This data is available for subscription by other Connex applications, and you can quickly verify and visualize the data using the RTI 'Admin Console' utility that comes with RTI Connex, which can be run from the 'Tools' tab of the RTI Launcher. The following steps show how to view the published vehicle data using Admin Console:

1. Make sure that the Unreal Engine project is running.
2. Open the RTI Launcher and click on the **Tools** tab.

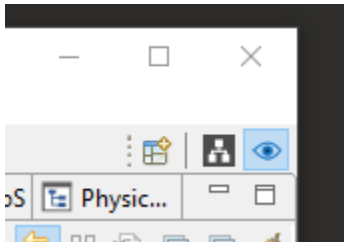


3. Click on **Admin Console**

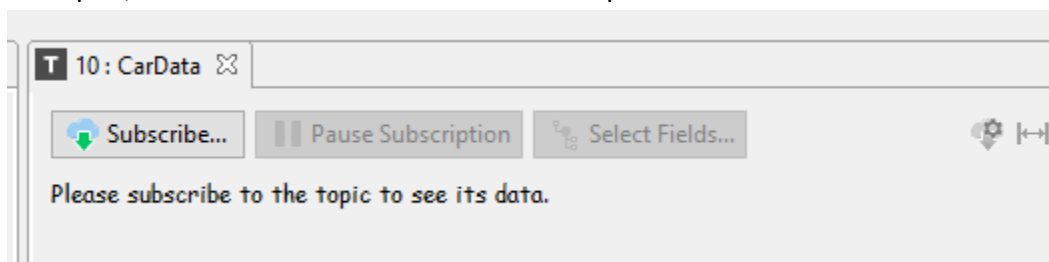
4. A few seconds after the Admin Console opens, it will automatically refresh the tree on the left panel and show the Domain 10 element. Unfold it and you will see the CarData element.



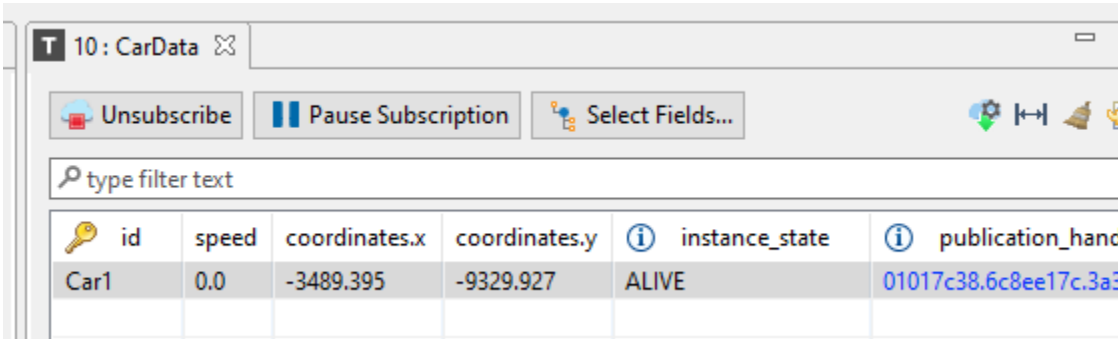
5. Admin Console has 2 viewing modes: Administration and Data Visualization, which are selectable in the upper-right corner of the application. Ensure Data Visualization mode is selected by clicking on the eye-shaped icon:



6. Click on the 'CarData' element under the 'Domain 10' entry in the 'DDS Logical View' tab in the upper left area of Admin Console. In the center viewspace, a tab for '10: CarData' will open, with a button to subscribe to this topic. Press this button.

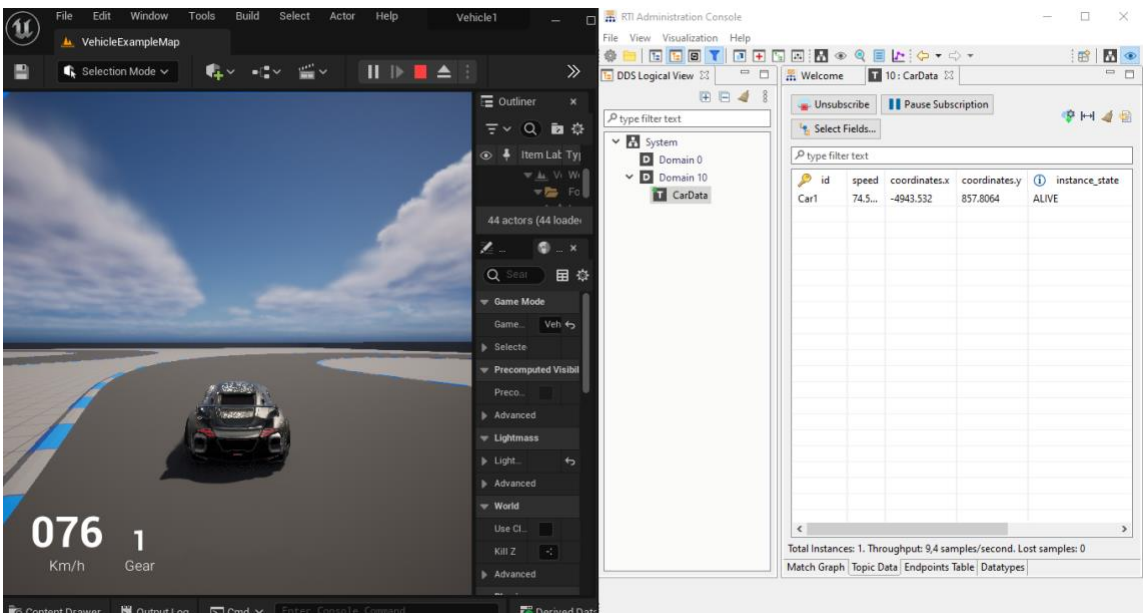


- A 'Create Subscription' dialog box will appear. Press 'OK'.
Admin console is now subscribed to and displaying the data from the CarData topic.



Note: The <0, 0> coordinate is right in the center of the circuit, so it is normal to have negative values.

- Split your screen in two halves, so that you can see the Unreal Editor on the left-hand side and the Admin Console on the right hand side



- Drive the car and see how the values change in real time!