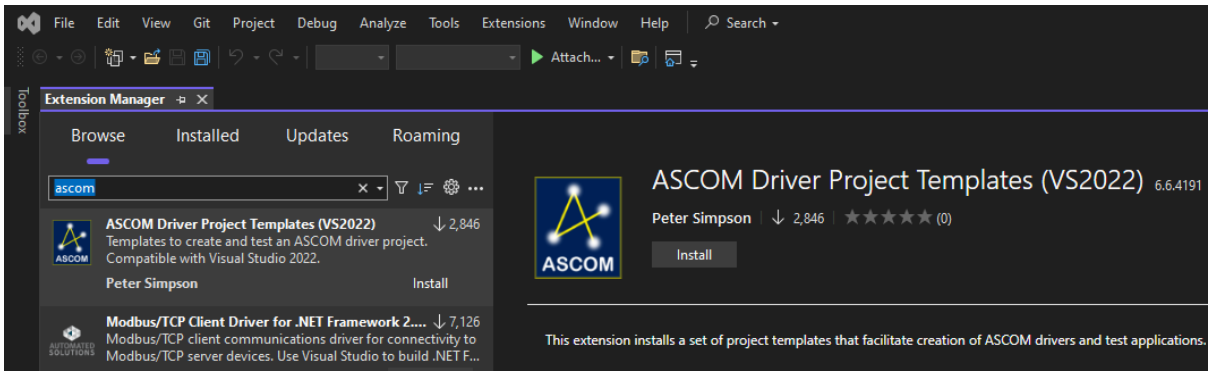
How to Make an ASCOM Driver

Obtain the Project Templates

The driver templates are available as a free Visual Studio Marketplace extension that can be installed via the dialogue shown by the Visual Studio Extensions / Manage Extension menu.

Enter the text **ascom** into the Browse search box to display the template installer:

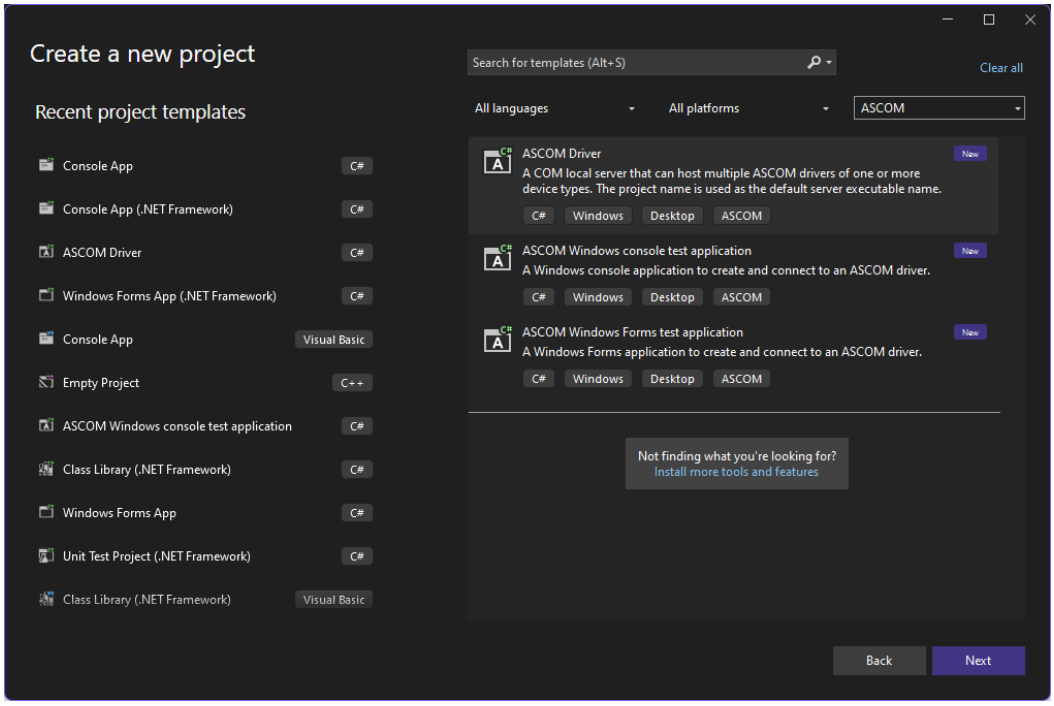


Click the Install button to schedule the templates for installation, which will happen after you close Visual Studio.

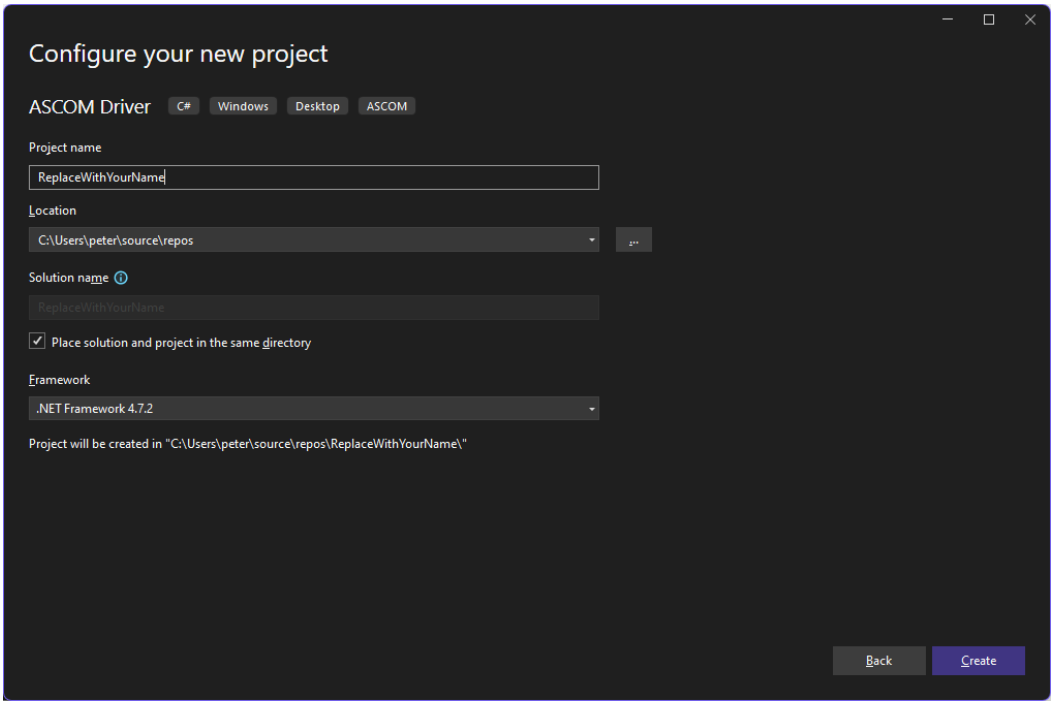
Create a starter project

Start Visual Studio and select Create a new project from the Get Started dialogue.

Click the dropdown that says All project types and select ASCOM from the list:

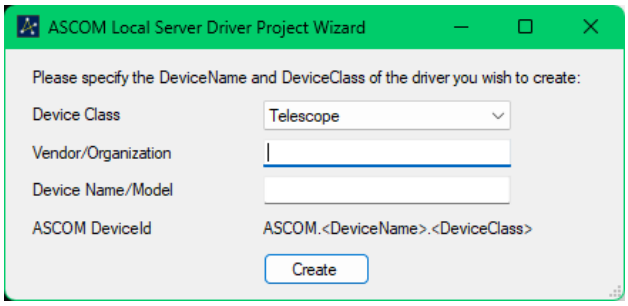


Select the ASCOM Driver entry and click Next.



Provide your own project name and change the project location if required then click Create.

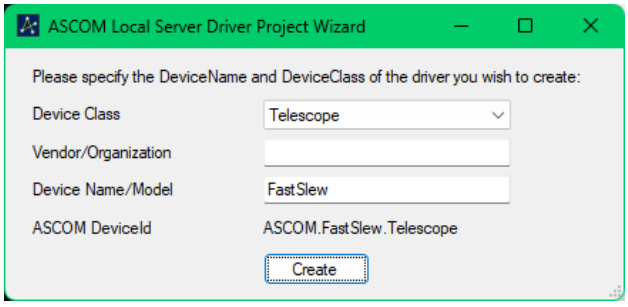
The ASCOM project customisation dialogue will now appear where you can set the basic parameters for your project.



Select the device type from the Device Class dropdown and enter a simple name for your device in the Device Name/Model field.

The device name must not contain spaces and must be short and unique for your device. This is not the description that the user will see in the Chooser, it will become part of the ASCOM DeviceId (actually the COM ProgID) that is used behind the scenes to identify your driver.

The Vendor/Organization field is optional.



Click the Create button to build the project.

Working with the project

Once the project has been generated you will see a read-me screen that describes how the project is laid out. Build the project, which should have 0 errors to confirm that it has been created correctly.

There is a lot of useful information in the read-me, particularly sections on how to partition functionality between driver class instances and the shared hardware class that all driver instances use to communicate with the device.

Clients connecting to your driver each get a unique instance of your driver class ***but there will only ever be one instance of the hardware class that is shared by all clients***. You should only add bespoke code to the driver class when the response is expected to be different for each client. Broadly speaking, most of your bespoke code will go in the shared hardware class.

You can now start adding your own functionality. The best overall approach is:

- Implement InterfaceVersion, returning the interface version number that your driver supports. E.g. if your device implements IRotatorV4, InterfaceVersion should return 4.
- Implement Connected so that you can talk reliably to your hardware. While one of the more complex areas, this is best tacked first. If your hardware doesn't support concurrent queries / actions you will need to serialise concurrent incoming requests so that the device only has to deal with one activity at a time.
- Implement the mandatory non-functional Properties such as Name, Description etc.
- Implement the mandatory functional properties for your device e.g. IsMoving for a Focuser device and Slewing for a Telescope device.
- Implement any mandatory methods for your device e.g. StartExposure for a camera driver.
- Implement optional properties and methods as required.

Conform is your friend!

The Conform Universal application is designed to help driver developers implement features effectively. It can assess the operation of every property and method in all ASCOM interfaces and confirm whether they operate in accordance with the ASCOM interface specifications.

Conform is designed to be robust and to operate in a development environment where interfaces may not be fully implemented or may be unreliable.

The latest version of Conform Universal can be found in this link: Conform Universal.