How to create an installer for a Windows service app (.NET Framework) using Visual Studio

Windows Services are applications that run in the background without displaying any user interface. They are a great option for performing tasks without disrupting the user's workflow on the machine.

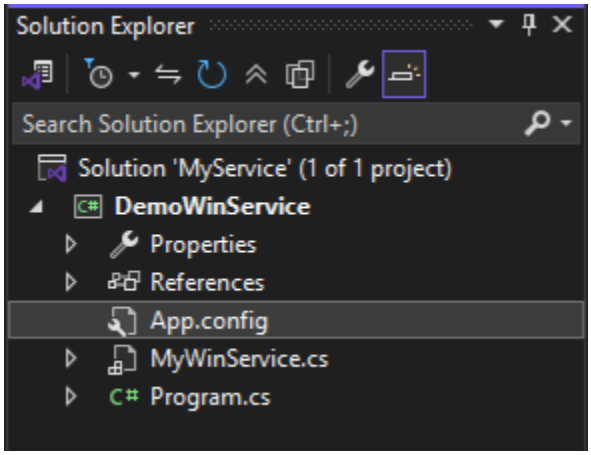
In this article, we'll guide you through:

- Creating and configuring a Windows Service targeting .Net Framework using Visual Studio.
- Building an installer for the service using Visual Studio. Additionally, we'll introduce a simpler alternative using Advanced Installer.

How to Create the Windows Service in Visual Studio?

First, let's walk through the steps to create the Windows Service project in Visual Studio:

- Go to File → New → Project.
- Select the Windows Service (.NET Framework) project template.
- In the configuration dialog, set a name for the service and choose the appropriate .NET Framework version.
- Once the project is created, you should see it in the solution explorer.



Configure the Windows Service in Visual Studio?

After creating the project, the next step is to add the logic for the service. The service main functionality is defined in the MyWinService.cs file. Here you can specify what happens when the service starts, stops, pauses, etc.

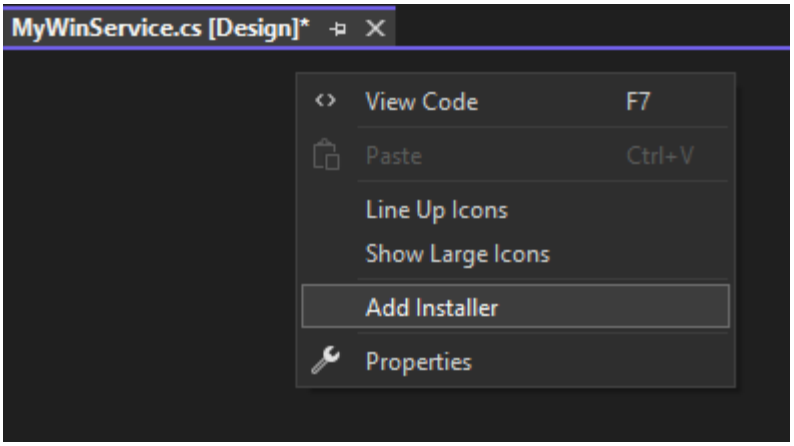
```
namespace MyService
{
    public partial class MyWinService : ServiceBase
    {
        public MyWinService()
        {
            InitializeComponent();
        }

        protected override void OnStart(string[] args)
        {
              //Method called by the OS when the service starts
        }

        protected override void OnStop()
        {
             //Method called by the OS when the service stops.
        }
       // You can also override OnPAuse(), OnContinue(). etc.
    }
}
```

The Program.cs is used to start the service, it serves as the entry point. Its main role is to create an instance of the service and register it.

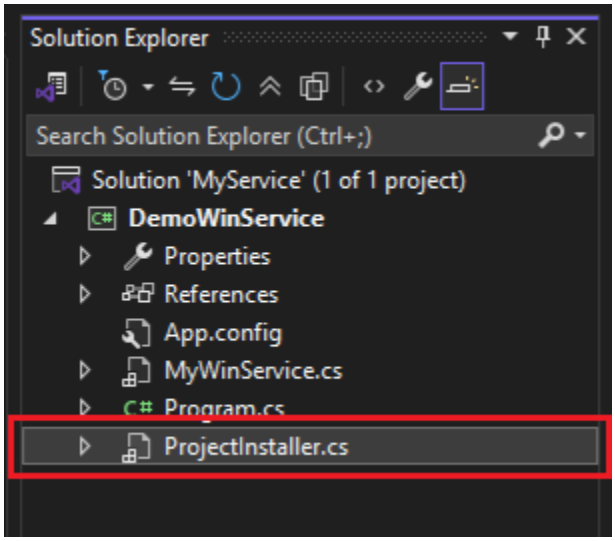
Add Installation Components to the Service

Once the service's core logic is created, we need to add installer components to the project. They provide the metadata and the logic required by the Windows operating system to correctly manage the service:

- Open MyService.cs
- Right-click in the designer view → Add Installer.

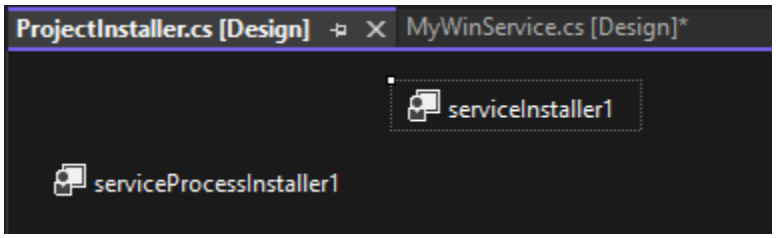


- This action will automatically generate a new file called ProjectInstaller.cs.



The ProjectInstaller.cs file is meant for the services installation, not for the service's runtime logic. It contains two components (visible if you double click on it):

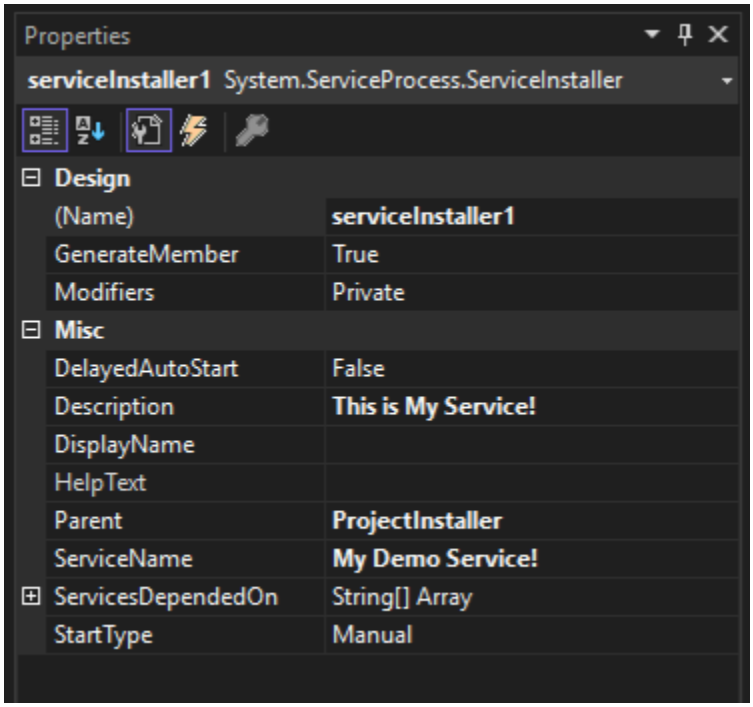
1. serviceInstaller - provides information about the service (name, description, start type).
2. serviceProcessInstaller - defines the account under which the service runs.



Configure the Installer Components

After adding the installer components, it's time to configure how the service will run and be registered. This involves setting properties for *serviceInstaller* and *serviceProcessInstaller* components. For the service properties:

1. Right-click on *serviceInstaller1* and select Properties.
2. Here you can find some properties that can be set. For example, you can change the start type of your service.

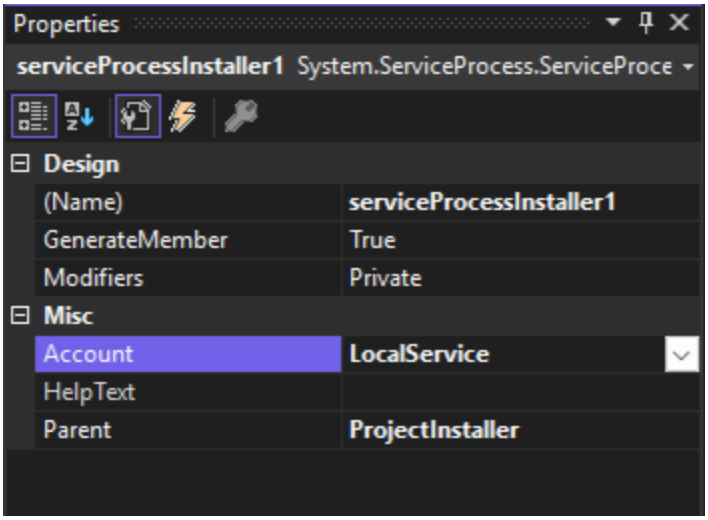


Here are the changes that we've made for our example service:

- We set the StartType to *Manual* so that the service starts manually. You can also change it to Automatic if you want it to start with the operating system at system start-up.
- The service name is *My Demo Service!*. This property is used by the system to identify the service.
- We've added a description to our service - *This is My Service!*. This description will appear in the Services Management console, providing more context about the service purpose.

To configure how the service will run:

- Select serviceProcessInstaller and go to the Properties section.
- Here, let's change the Account property from User (which requires specifying username and password) to LocalService to ensure the system will not prompt for user credentials.



Add the Microsoft Visual Studio Installer Projects extension

Once your service is configured with its installation components, you have to create the installer package for your service application. To do it, you will need to use the Setup project template which is available only if you have the Microsoft Visual Studio Installer Projects extension installed.

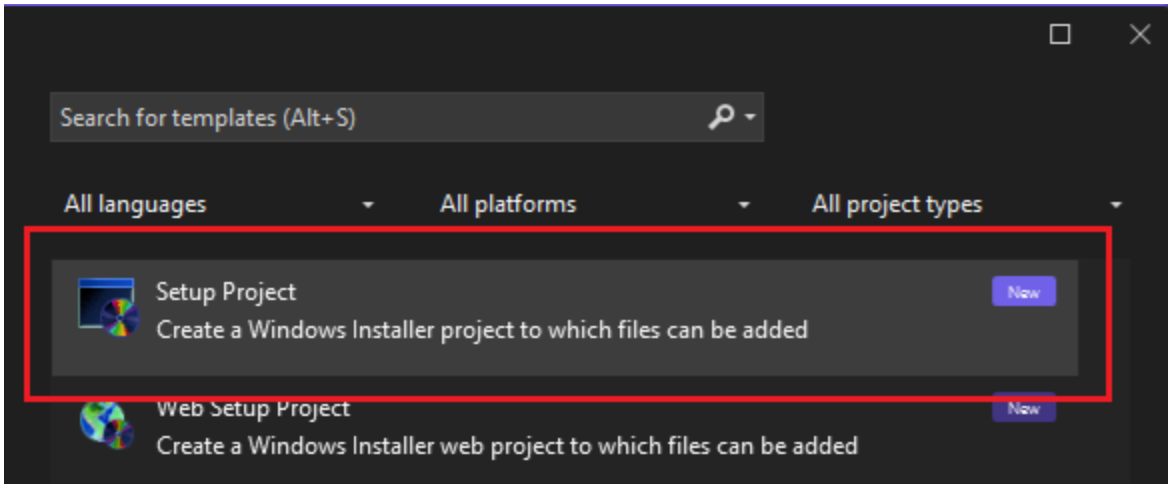
There are two available options to add the Microsoft Visual Studio Installer Projects extension:

1. In **Visual Studio**, go to *Extensions → Manage extensions*. Look for the extension in the Online section and download it. It will be automatically installed after you close the IDE. When you reopen Visual Studio, the extension will be ready to use.
2. Go to the **Visual Studio Marketplace**, download the extension and install it. Make sure that the extension version is suitable for your Visual Studio version..

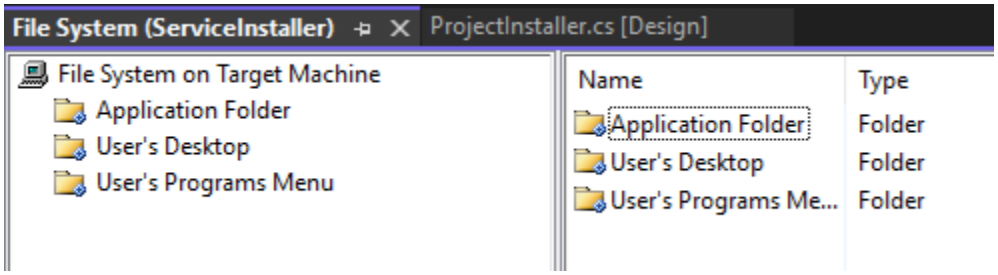
Add the Setup Project to your Service Application

Once the extension is installed, here's how you can add the Setup project:

- Right-click on the project solution → *Add → New Project.*
- Select the *Setup Project* template from the list of templates and give it a name. The name of my Setup Project is ServiceInstaller.



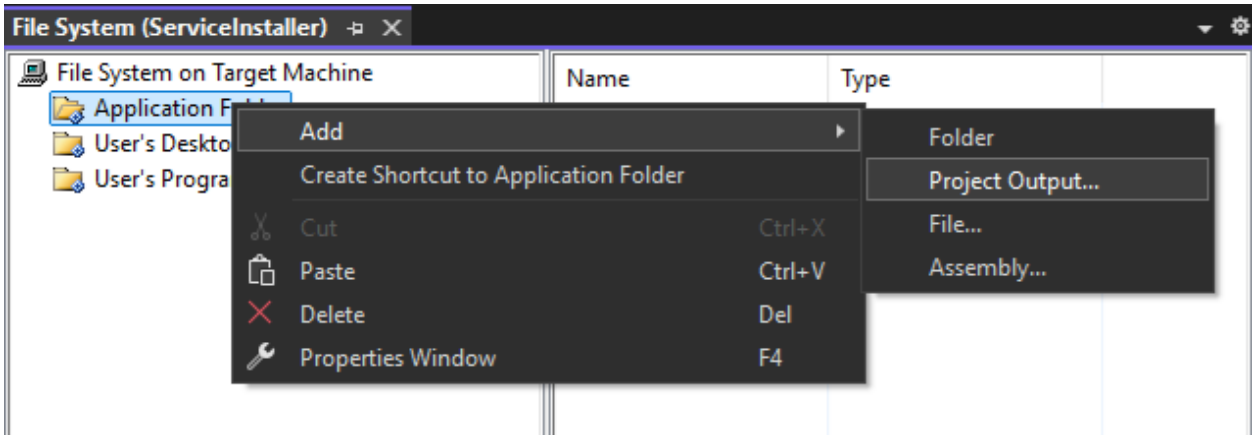
- Once the setup project is created, you should see it in Solution Explorer.
- To see how the Setup Project is structured, right-click on the *Setup project* → *View* → *File System*.



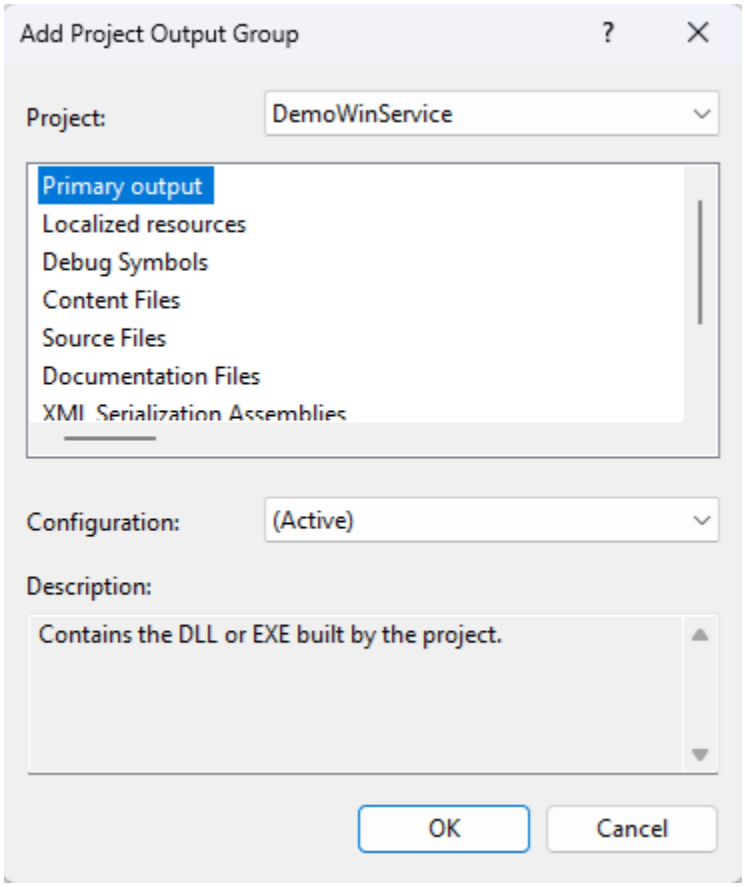
Add the Project Output to the Setup Project

Next, we need to add a project output for the Setup project.

- Right-click on the *Application Folder* in the File System → *Add* → *Project Output.*



- Choose the *Primary output* option from the opened dialog.

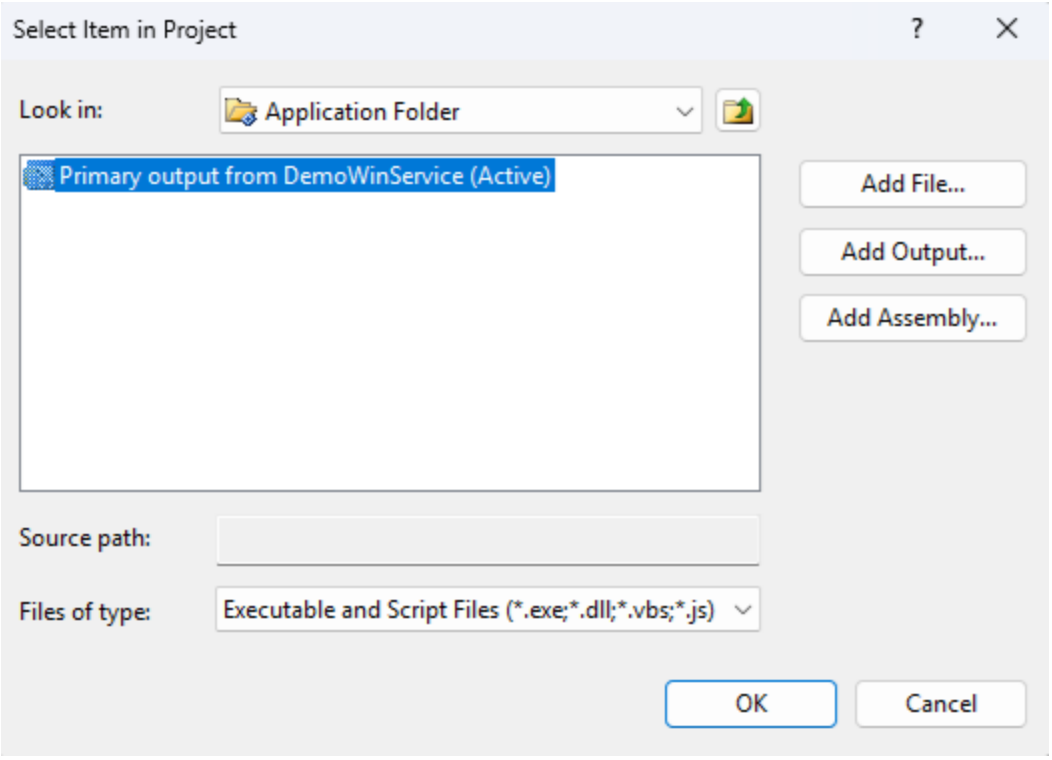


Add a Custom Actions to Install the Service

To register the service with the Windows Service Control Manager (SCM) during installation, you need to add a custom action that points to the service's Primary output. Thus, the installer executes the service's built-in installation logic (from ProjectInstaller.cs) which handles the service registration.

> If you're looking for an easier alternative to create the installer package for your service, you can use the Advanced Installer tool. It automatically handles the service registration so you don't need to add any custom action.

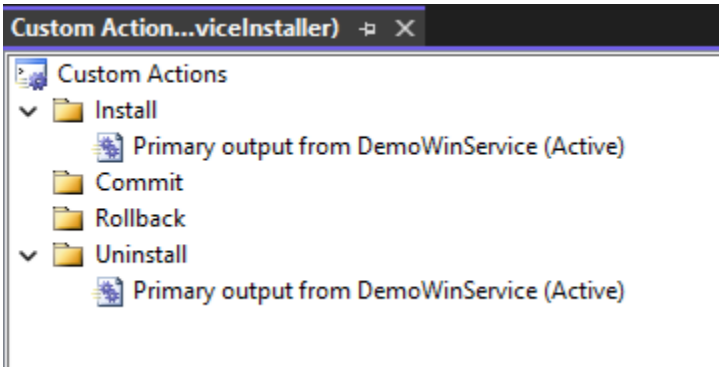
To add the custom action:
1. Right-click on ServiceInstaller → View → Custom Action.
2. Then, right-click on the Install folder under Custom Actions and select *Add Custom Action*.
4. In the opened dialog, look in the Application Folder and select Primary output.



What if you need to uninstall the service?

Similarly, to unregister the service when the application is uninstalled, you should also add a custom action to the Uninstall step.

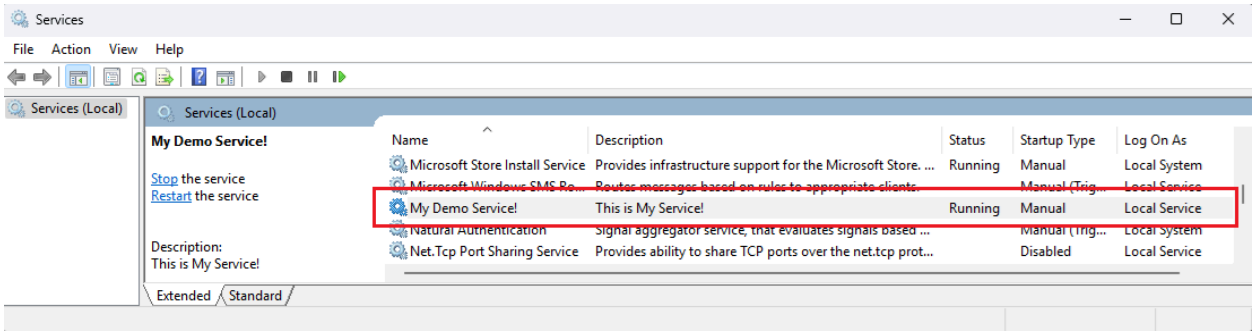
- Go to the **Uninstall** folder from Custom Actions and follow the same steps as you did for the Install step.
- After this, you will see your custom actions created like in the image below.



Build and Install the Setup Project

Now that the Setup Project is configured, it's time to build it to generate the installer. Then, you should find it in the Debug or Release folder (depending on the build mode) of the Setup Project.

Finally, run the setup and after the installation, look for your service in the Service Control Manager. You should identify the service by the name set along with the description.



We can identify the service (My Demo Service!) along with the description we set. The service is running as we started it manually. If you set the Start type to automatically, it should be running without you having to start it.

How to Create the installer using the Advanced Installer tool?

For a more streamlined and user-friendly experience when creating installers for service applications, you can consider using a dedicated GUI packaging software like Advanced Installer.

This tool can significantly simplify the process of service customizations. You simply add your compiled service executable to an Advanced Installer project and then take advantage of its configuration options directly within its intuitive graphical user interface.

For a visual guide on how to create an installer for a Windows service using the Advanced Installer extension for Visual Studio, you can refer to this tutorial:

https://www.youtube.com/watch?v=jNbI2r5YNYs&ab_channel=AdvancedInstaller%26PacKit

Creating the service Installer with Visual Studio vs Advanced Installer

Feature	Visual Studio	Advanced Installer
GUI-based configuration	Limited	Full-featured

Automatically service register/unregister	No - requires custom actions	Yes - handled automatically
Control over service operation parameters	Very limited - must be handled via custom code	Full control via GUI
Supports additional UI (dialogs, branding)	No (limited control)	Yes (custom UI dialogs, themes)
Suitable for complex service configuration	Limited	Highly suitable

Conclusion

In addition to creating a service, Visual Studio offers the possibility to create an installer for that application service. Things look quite simple if you want to install a service which doesn't require a complex setup, but it could become difficult if the service requires many advanced configurations.

FAQ section

1. What project template is used in Visual Studio for a service targeting .NET Framework?
2. What are the installer components of a service?
3. How do I configure a Windows Service?
4. Do I need to add custom actions to register a service at install time?
5. Does Advanced Installer automatically register a service at install time?