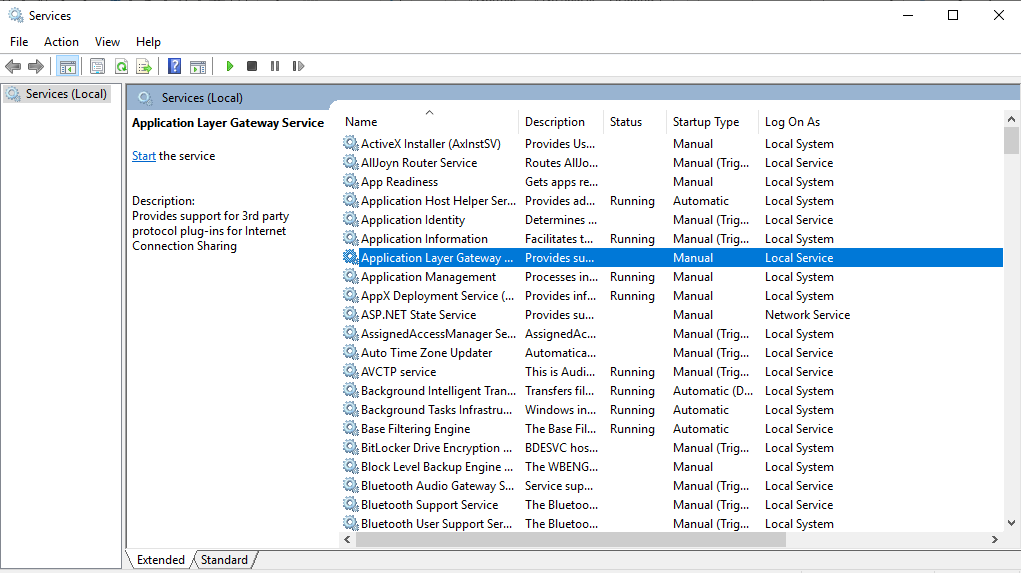
Windows Service

# Overview

Microsoft Windows services are long-running executable applications that run in the background, similar to daemons found in Unix-like OSes. Windows Services can be found in Windows Services Manager.

Windows Services Manager: Windows + R => services.msc => Enter.



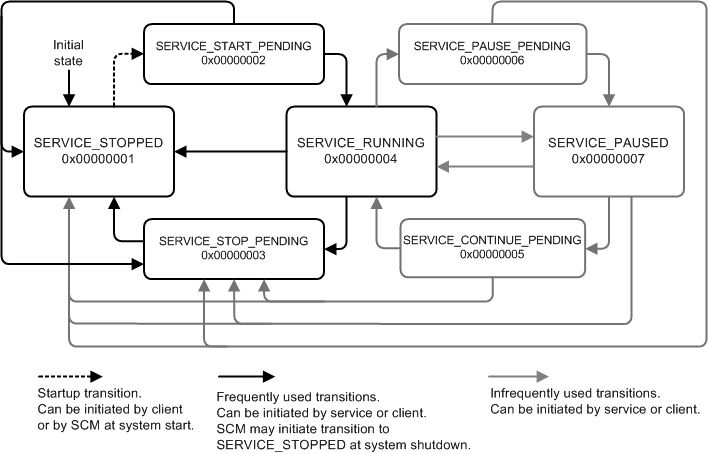
# Windows Service vs Other Windows Application

These are several key differences:

* The compiled executable file of the service must be installed on the host machine before it can be start.
* You cannot debug Windows service by normal means. You must install and start your service, and then attach a debugger to the service's process.
* Windows service applications run in their own security context and are started before the user logs on. Use caution when providing the service with accounts having more permissions and privileges than it needs.

# Service Lifetime

Below are the possible state transitions of a Windows Service.



A service is responsible for reporting changes in its state to the service control manager (SCM). Service control programs and the system can find out the state of a service only from the SCM, so it is important that a service report its state correctly.

# Service Application Programming Architecture

Windows Service applications are based on a class that inherits from the System.ServiceProcess.ServiceBase class. You override methods from this class and define functionality for them to determine how your service behaves.

The main classes involved in service creation are:

* System.ServiceProcess.ServiceBase — You override methods from the ServiceBase class when creating a service and define the code to determine how your service functions in this inherited class.
* [System.ServiceProcess.ServiceProcessInstaller](https://docs.microsoft.com/en-us/dotnet/api/system.serviceprocess.serviceprocessinstaller) and [System.ServiceProcess.ServiceInstaller](https://docs.microsoft.com/en-us/dotnet/api/system.serviceprocess.serviceinstaller) —You use these classes to install and uninstall your service.

## Defining Your Service's Behavior

In your service class, you override base class functions that determine what happens when the state of your service is changed in the Services Control Manager. The ServiceBase class exposes the following methods, which you can override to add custom behavior.

|  |  |
| --- | --- |
| Method | Override to |
| OnStart | Indicate what actions should be taken when your service starts running. You must write code in this procedure for your service to perform useful work. |
| OnPause | Indicate what should happen when your service is paused. |
| OnStop | Indicate what should happen when your service stops running. |
| OnContinue | Indicate what should happen when your service resumes normal functioning after being paused. |
| OnShutdown | Indicate what should happen just prior to your system shutting down, if your service is running at that time. |
| OnCustomCommand | Indicate what should happen when your service receives a custom command. For more information on custom commands, see MSDN online. |
| OnPowerEvent | Indicate how the service should respond when a power management event is received, such as a low battery or suspended operation. |

# Creating and running a service

This process is partially based on the [official documentation](https://docs.microsoft.com/en-us/dotnet/framework/windows-services/walkthrough-creating-a-windows-service-application-in-the-component-designer).

The service we’ll be creating will have the following flow:

* The method OnTimer will be called every 10 seconds (configurable), the System.Timers.Timer object will be started at Service start.
* The Timer object is disposed if paused or stopped, and reinitialized if continued.
* All configurable settings are found in **BackupToJsonService.exe.config** file, being in the same folder as the Service executable when compiled.

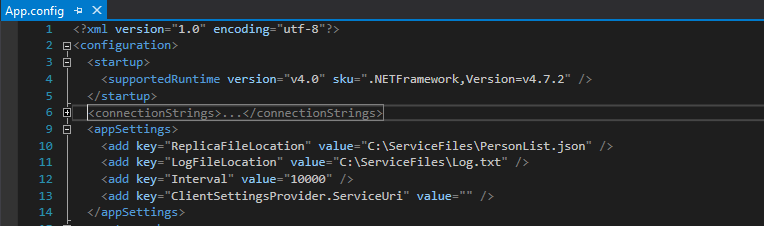


Figure . Available configurations

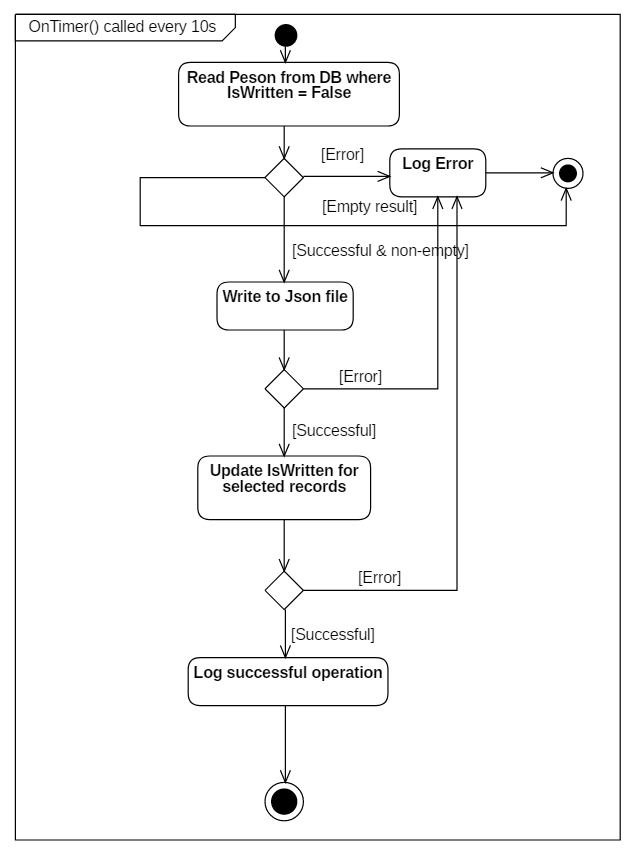
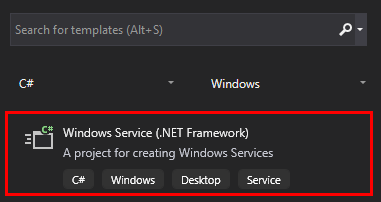


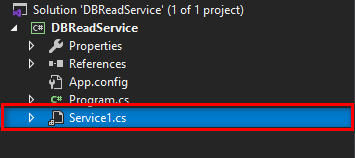
Figure . Service main flow

## Create Windows Service Project

Create the Windows Service Project Template in Visual Studio.



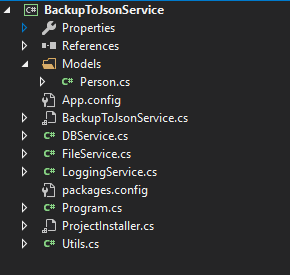
Rename the pre-created Service1.cs to the name **DBReadService**.



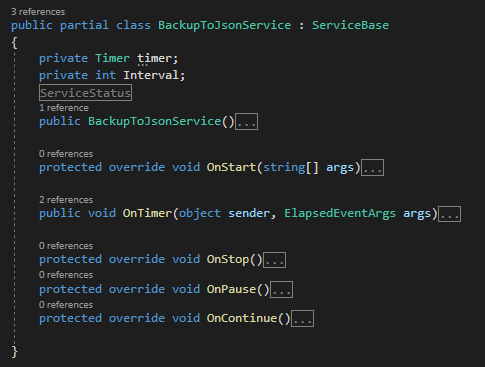
In the **Design** tab, select **Properties** from the shortcut menu. From the **Properties** window, change the **ServiceName** value to **BackupToJsonService,** this will be the display name of our service when loaded and run (Remember to save the file).

## Adding features to the service.

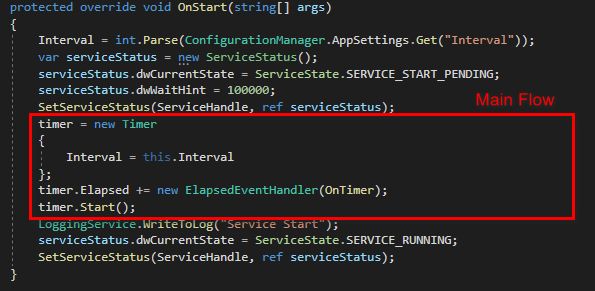
The project’s structure:



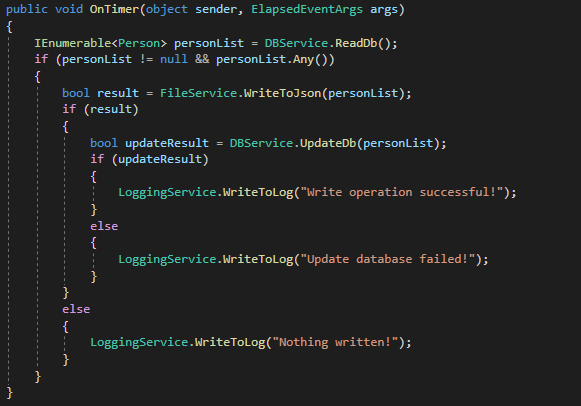
The BackupToJsonService class will look like this:



The method OnStart() will be invoked on service start, will describe the main logic of the service.

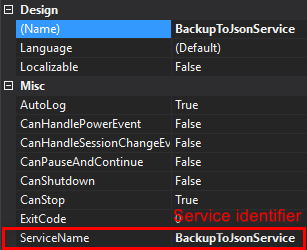


OnTimer event-handler will run between every interval:



The other supporting classes and functions can be found in the source code: <https://github.com/popemkt/BackupToJsonService>

As mentioned earlier in this document, the service must implement an installer to be able to work properly. Right click BackUpToJsonService.cs => View Designer => Right-click menu: Add installer. A file named ProjectInstaller.cs will be added to the project. To change some service settings: Right click BackUpToJsonService.cs => View Designer => Right-click menu: Properties. We will rename the service:



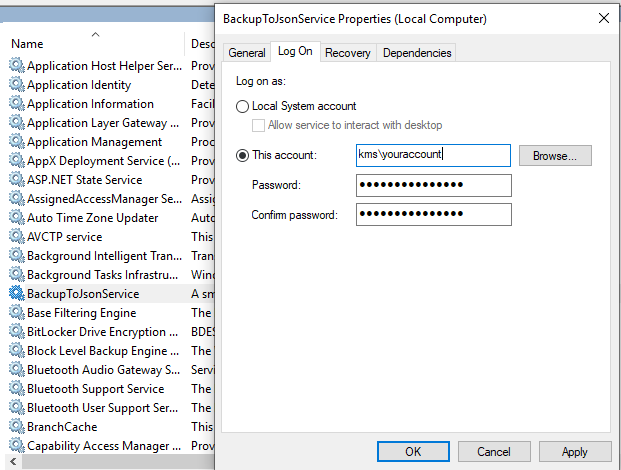
## Install the service

Build the project then using **Developer Command prompt for Visual Studio** with **Administrator** privileges, cd to the folder containing the built binaries, usually: ..ProjectFolder/bin/Debug/ :

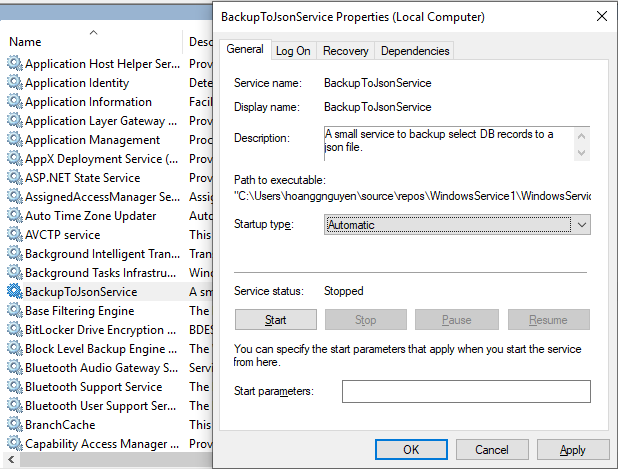


After installation, provide the suitable credentials for the service to login as in services.msc.

Right-click BackupToJsonService => Properties => Log On Tab:

(For uninstallation, cd to the same folder containing the built binaries, run: installutil /u BackupToJsonService.exe, must also use Administrator privileges) 

Then manually set start-up type and start the service:

  
The Service is now running and can be stopped – continue on demand.