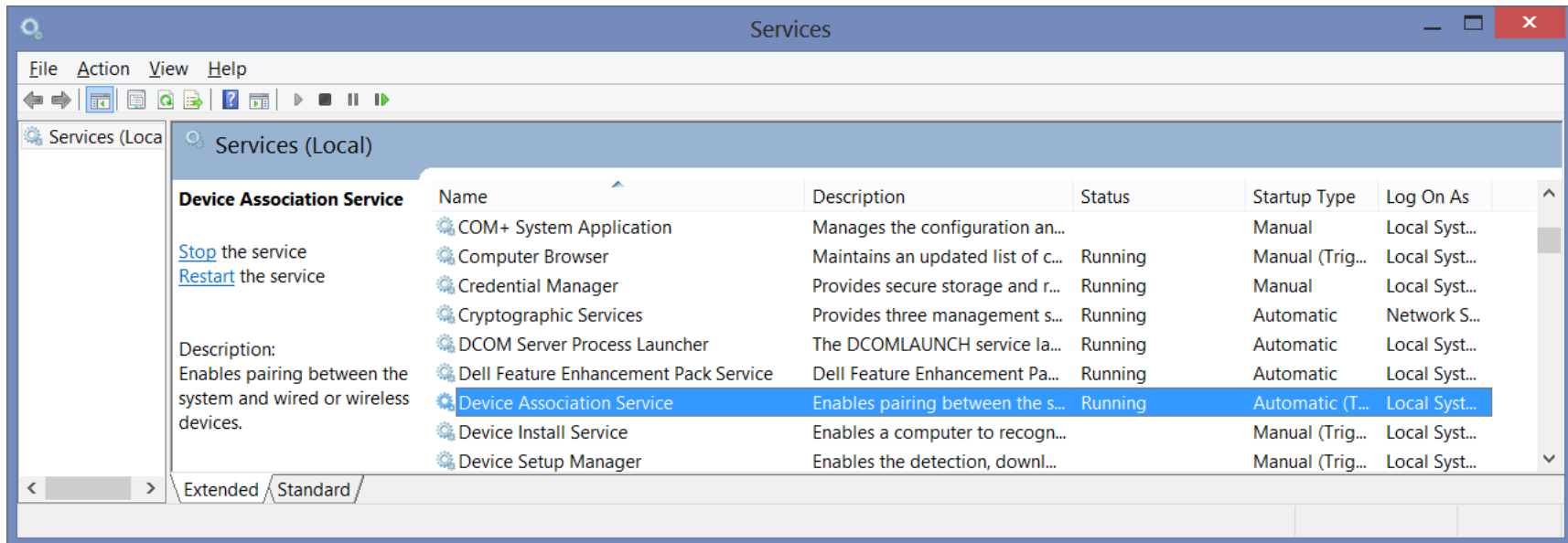


# Windows Services

# What is a Windows service?

- Windows services are applications that run outside of any particular user context.
  - They don't require a user to be logged in in order to do their work.
  - They run in their own Windows sessions .
  - And they generally run in a higher-powered security mode than most users do.
- Typically these services provide system-level support
  - and runs without human intervention
- Windows services can be started in 3 different ways:
  - By Windows when the system boots up
  - By a user, using the Services utility found within the Administrative Tools grouping in Control Panel
  - By an application, via calls to the Windows Service Control Manager.

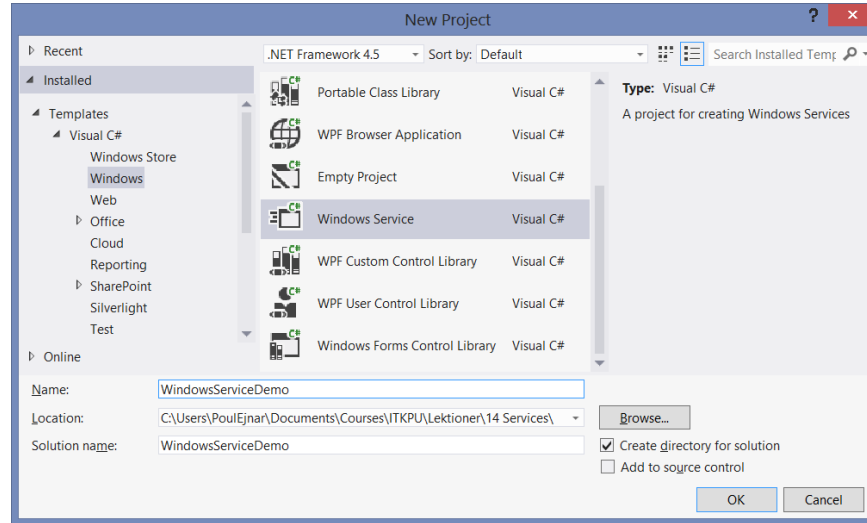


# Windows Services

- Are similar in concept to a Unix daemon.
- Many appear in the processes list in the Windows Task Manager
  - often with a username of SYSTEM, LOCAL SERVICE or NETWORK SERVICE
  - or run through svchost.exe as DLLs loaded into memory.
- A Windows Service Application is created as .NET-enabled project that will create an .exe file when built and inherits from the ServiceBase Class.
  - You will normally use the dedicated Wizard in VS.
- Projects containing Windows services must have installation components for the project and its services.
  - This can be easily accomplished from the Properties window.

# Creating a Windows Service Project


- Create (Add) a new project by use of the Windows Service Wizard:



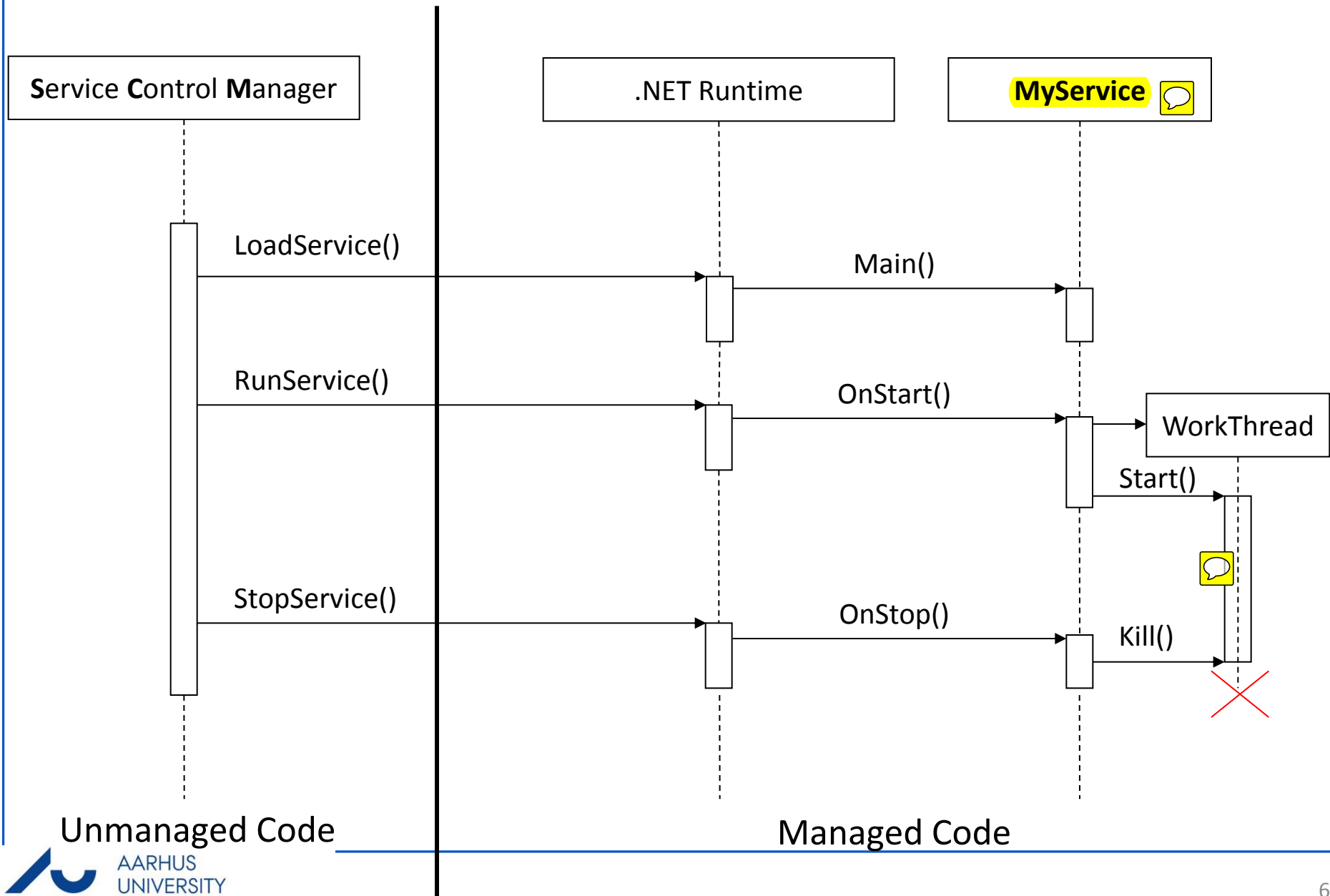
- This Wizard will provide you with a default implementation of main() that is suitable for many projects.

```
static void Main()
{
    ServiceBase[] ServicesToRun;
    ServicesToRun = new ServiceBase[]
    {
        new Service1()
    };
    ServiceBase.Run(ServicesToRun);
}
```

# Implementing Windows Services

- The Wizard will also provide a Service1 class that inherits from ServiceBase.
- You just add overrides for the relevant functions inherited from ServiceBase.
  - All functions incl. main must return within 30 seconds. 
- To create a functional service you must:
  - Set the **ServiceName** property.
  - Override and specify code for the **OnStart** and **OnStop** methods to customize the ways in which your service behaves.
  - Create the necessary installers for your service application.

# SCM $\leftrightarrow$ Service Interactions

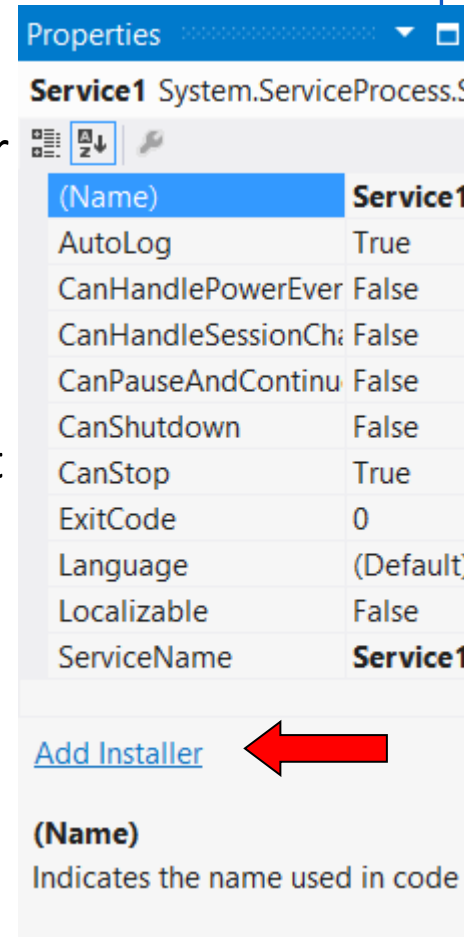


# Possible Overrides

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.


# Create Installers

- To add an installer to your service project you must:
  - Open (or select) the design view for the Service, and then click the background of the designer to **select the service** itself.
  - Then you can click on the **Add Installer** link in the Properties window.
- A component class containing two installers is added to your project.
  - one installer for your service
  - one installer for the service's associated process.
- You can use the properties window to configure these installers. Access Design view for **ProjectInstaller**, and select the appropriate installer.
- For the ServiceInstaller set:
  - the **ServiceName** property and
  - the **StartType** property.
- For the **serviceProcessInstaller** set:
  - the **Account** property to the appropriate account.





# Service Account Types

Member name	Description
LocalService	An account that acts as a non-privileged user on the local computer, and presents anonymous credentials to any remote server.
LocalSystem	An account that has a high privileged level. <i>(Use with caution)</i>
NetworkService	An account that provides extensive local privileges, and presents the computer's credentials to any remote server. 
User	An account defined by a specific user on the network. <i>Specifying User for the ServiceProcessInstaller.Account member causes the system to prompt for a valid user name and password when the service is installed, unless you set values for both the Username and Password properties of your ServiceProcessInstaller instance.</i>

*When you have configured the installers you are ready to build the project.*

# Create A Setup Project For The Service

- Select **New Project**.
  - select the **Setup and Deployment Projects** folder.
  - In the **Templates** pane, select **Setup Project**.
  - Name the project.
- Add the output from the Windows Service project to the setup project
  - In Solution Explorer, right-click <name>Setup, select Add, then choose Project Output. MyNewService is selected in the Project box.
  - The Add Project Output Group dialog box appears
    - Select Primary Output, and click OK.



In  
VS2010

# Add A Custom Action To Install The Setup Project

- Right-click the setup project, select **View**, then choose **Custom Actions**.
- In the **Custom Actions** editor, right-click the **Custom Actions** node and choose **Add Custom Action**.
  - The Select Item in Project dialog box appears.
- Double-click the **Application Folder** in the list box to open it, select **Primary Output from <name>Service (Active)**, and click OK.
- The primary output is added to all four nodes of the custom actions — **Install, Commit, Rollback, and Uninstall**.
- Build the project.
- To install the Windows Service
  - Right-click the setup project in the Solution Explorer and select Install.

In  
VS2010

# How to: Install and Uninstall Services

1. On the Windows Start menu/Screen, choose “Visual Studio Command Prompt”.
2. Access the directory in which your project's compiled executable file is located.
3. Run InstallUtil.exe from the command prompt with your project's output as a parameter.

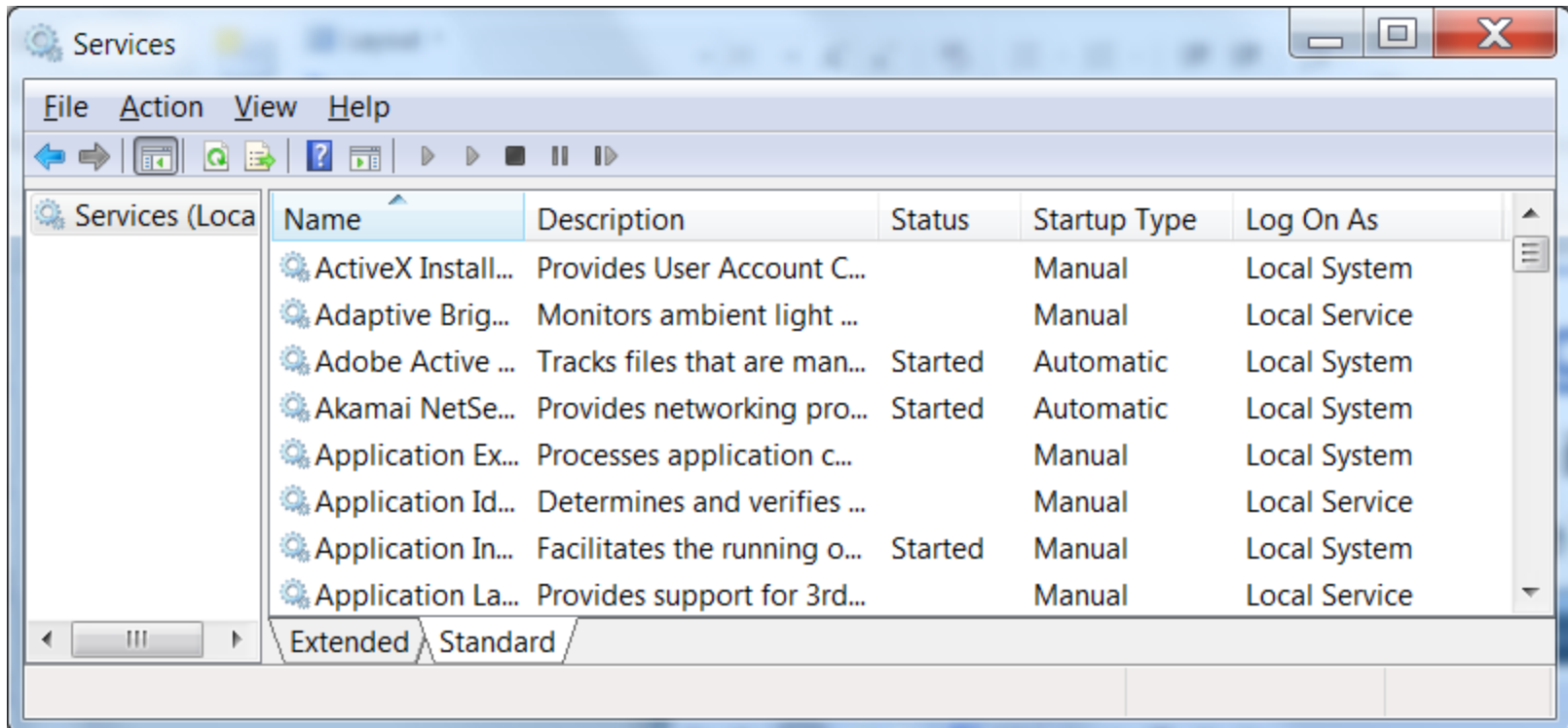
```
installutil yourproject.exe
```

- To uninstall your service manually


```
Installutil /u yourproject.exe
```

# Managing Services

- Once a service is installed, it can be managed by:
  - launching "Services" from the Windows Control Panel → *Administrative Tools*.
  - Typing "Services.msc" at the command prompt..
  - Or by use of the SC command at the command prompt.



# Debugging Windows Services

- You cannot press F5 to run your service project!
- To ease debugging it is highly recommended to split your service in 2 (or more) layers:
  - A top layer as thin as possible that implements the required service interface.
  - A layer that implements the functionality of the service, and can be tested from a normal .Net project before your start to test the service project.
- To debug a Windows Service:
  - Build the service project
  - Install it
  - Start the service, and then
- In Visual Studio, with your project loaded, select the Debug | **Attach to Processes menu item**, displaying the Processes dialog box.
  - Select the Show system processes checkbox so that the Available Processes list includes running services.
  - In the Available Processes list, select your service entry, and then click Attach.
  - On the Attach to Process dialog box, make sure that the Common Language Runtime option is selected, then select OK to accept. Then you can close the Processes dialog box.
- Set a breakpoint in your code at the location you'd like to test.
  - Trigger the breakpoint by taking the necessary action.
  - At this point, you can single-step through the code, as you would with any other application.
  - Press F5 to continue running when you're finished with your testing.
- When you're finished debugging, detach the debugger from the running process using the Debug | Processes menu.
- Use the Debug | Stop Debugging menu item to end the debugging session. 

# CONTROLLING A SERVICE FROM AN APPLICATION

# The ServiceController

- You can use the ServiceController class to connect to and control the behavior of existing services.
- When you create an instance of the ServiceController class, you set its properties so it interacts with a specific Windows service.
  - You can then use the class to start, stop, and otherwise manipulate the service.



# Connecting to the Service

- Use `ServiceController.GetServices()` to get a list of all services running on the machine.
- Iterate through the list and **search for the service name you need.**

```
static void Main(string[] args)
{
    ServiceController[] scServices;
    scServices = ServiceController.GetServices();
    foreach (ServiceController scTemp in scServices)
    {
        if (scTemp.ServiceName == "Simple Service")
        {
            // Display properties for the Simple Service sample
            // from the ServiceBase example.
            ServiceController sc = new ServiceController("Simple
                                                         Service");
        }
    }
}
```

# Custom Commands

- You can send custom commands to the service, but must be int between 128 and 255.

```
public enum SimpleServiceCustomCommands { StopWorker = 128,
RestartWorker, CheckWorker };

static void Main(string[] args)
{
    ServiceController sc = new ServiceController("Simple Service");
    ...
    sc.ExecuteCommand((int)SimpleServiceCustomCommands.StopWorker);
    sc.ExecuteCommand((int)SimpleServiceCustomCommands.RestartWorker);

    String[] argArray = new string[] {"ServiceController arg1",
                                     "ServiceController arg2"};
    sc.Start(argArray);
}
```

# References

- Creating a Windows Service Application in C#  
<http://msdn.microsoft.com/en-us/library/y817hyb6.aspx>
- Windows services – Unmanaged reference  
[http://msdn.microsoft.com/en-us/library/ms685141\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms685141(VS.85).aspx)