Platforms, Frameworks & Libraries » .NET Framework » How To

C#, .NET, Win2K, WinXP, Windows, Visual Studio, Dev

Adding a description to a .NET Windows Service

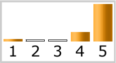
By **Andy Hopper**

Posted	: **6 Feb 2002**
Updated	: **6 Feb 2002**
Views	: **174,087**
Bookmarked	: **90 times**

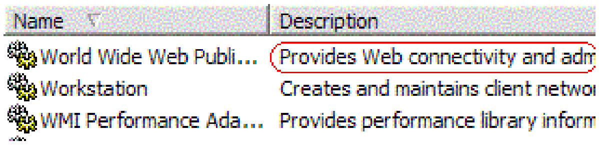
This article describes how to add a description for your .NET Framework Windows Service to the Services administration tool.

59 votes for this Article.

Popularity: 8.16 Rating: **4.61** out of 5



Download source files - 1 Kb



Name	Description
World Wide Web Publi...	Provides Web connectivity and adm
Workstation	Creates and maintains client networ
WMI Performance Ada...	Provides performance library inform

Introduction

Although the .NET Framework provides extremely robust Windows Service support through the classes available under the System.ServiceProcess namespace, for some reason the ability to specify your the description displayed in the Services control panel applet/MMC snap-in for your service was omitted. There exists an attribute class named ServiceProcessDescription, but it actually specifies what the Services MMC displays under the name column, and the Description column is left blank. This article will walk you through a low-level hack for adding a description by adding it directly to your service's registry key.

Most services keep their configuration information in the registry under the HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\ key. This is where the Service Control Manager (SCM) looks to get a list of services installed on a machine, and the Services control panel uses the SCM to list and modify the services. If you look under the key for a service, you'll see several entries, but we're interested in one in particular: the Description value. This is a REG_SZ (string) value, and this is where the SCM looks to get a service's description. We'll now take advantage of this arcane knowledge in the code below (you may download the source by clicking here):

```
 //This code should be inserted into your ProjectInstaller class' code

public override void Install(IDictionary stateServer)
{
  Microsoft.Win32.RegistryKey system,
    //HKEY_LOCAL_MACHINE\Services\CurrentControlSet
    currentControlSet,
    //...\Services
    services,
    //...\<Service Name>
    service,
    //...\Parameters - this is where you can put service-specific configuration
    config;

  try
  {
    //Let the project installer do its job
    base.Install(stateServer);

    //Open the HKEY_LOCAL_MACHINE\SYSTEM key
    system = Microsoft.Win32.Registry.LocalMachine.OpenSubKey("System");
    //Open CurrentControlSet
    currentControlSet = system.OpenSubKey("CurrentControlSet");
    //Go to the services key
    services = currentControlSet.OpenSubKey("Services");
    //Open the key for your service, and allow writing
    service = services.OpenSubKey(this.serviceInstaller1.ServiceName, true);
    //Add your service's description as a REG_SZ value named "Description"
    service.SetValue("Description", "This is my service's description.");
    //(Optional) Add some custom information your service will use...
    config = service.CreateSubKey("Parameters");
  }
  catch(Exception e)
  {
    Console.WriteLine("An exception was thrown during service installation:\n" + e.ToString());
  }
}

public override void Uninstall(IDictionary stateServer)
{
  Microsoft.Win32.RegistryKey system,
    currentControlSet,
    services,
    service;

  try
  {
    //Drill down to the service key and open it with write permission
    system = Microsoft.Win32.Registry.LocalMachine.OpenSubKey("System");
    currentControlSet = system.OpenSubKey("CurrentControlSet");
    services = currentControlSet.OpenSubKey("Services");
    service = services.OpenSubKey(this.serviceInstaller1.ServiceName, true);
    //Delete any keys you created during installation (or that your service created)
    service.DeleteSubKeyTree("Parameters");
    //...
  }
  catch(Exception e)
  {
    Console.WriteLine("Exception encountered while uninstalling service:\n" + e.ToString());
  }
  finally
  {
    //Let the project installer do its job
    base.Uninstall(stateServer);
  }
}
```

After you add the above code to your ProjectInstaller class, you should see a description for your service alongside your service's name after your service is installed. In a future article, we'll look at how we can add the description as a custom attribute and extend the ServiceInstaller class to add the description for us automatically.

License

This article has no explicit license attached to it but may contain usage terms in the article text or the download files themselves. If in doubt please contact the author via the discussion board below.

A list of licenses authors might use can be found here

About the Author

Andy Hopper

Andy Hopper is a senior software engineer with DocuSys, a company that supplies digital medical solutions. He primarily designs and implements multi-tier backend architectures for Medical Informatics applications.
Classically trained as an electrical engineer, Andy rediscovered his long-lost love with programming while working on a Master's degree project and changed careers in 1995 to become a self-taught software engineer. Andy realized his time with VB and ATL were at an end when he received the .NET Framework preview at the 2000 Professional Developer's Conference.
Andy has previously worked as a Resesarch Engineer at the Georgia Institute of Technology, a Senior Software Engineer at CyberCare Technologies, and a Lead Systems Architect for the American Cardiovascular Research Institute.

Flamebait: Andy thinks arrays should be 0-based.

Occupation:	Web Developer
Location:	United States

Discussions and Feedback

45 messages have been posted for this article. Visit **http://www.codeproject.com/KB/dotnet/dotnetscmdescription.aspx** to post and view comments on this article, or click **here** to get a print view with messages.

PermaLink | Privacy | Terms of Use
Last Updated: 6 Feb 2002
Editor: Chris Maunder