General Reading » Hardware & System » Windows Services License: The Code Project Open License (CPOL) VC6Win2K, Visual Studio, Dev

Posted: **7 Sep 2000**
Updated: **22 Feb 2008**
Views: **4,367,864**
Bookmarked: **591 times**

Start Your Windows Programs From An NT Service

By Xiangyang Liu 刘向阳

Make your MFC, VB and other Windows programs behave like NT services.

Download source code - 25.2 Kb

Introduction

Typically, a Windows service -- used to be called NT service -- is a console application that does not have a message pump. A Windows service can be started without the user having to log into the computer and it won't die after the user logs off. However, it is hard -- sometimes impossible -- to use many existing ActiveX controls within a console application.

On the other hand, MFC and VB applications are Windows applications, so using ActiveX controls in MFC or VB programs is extremely easy. It would be nice to make your MFC and VB programs run like a Windows service, so that:

- They will be started before the user logs onto the computer.
- They will keep running after the user has logged off.

It is possible to write a Windows service as a Windows program, but here I am proposing a much easier solution. I have included with this article the source code for a simple Windows service program that can start and shut down other programs. All you need to do is install this service and modify the INI file. Here are the advantages of using this simple Windows service:

- It can start up to 127 programs of your choice. The started programs behave like Windows services in that they will be running in the background without the user having to log into the machine.
- A user cannot kill the programs started by this service without proper privilege unless, of course, the machine is shutdown.
- You can test and debug your programs outside of the Windows service. For example, you can run your programs in the Visual Studio debugger and step into the source code to find bugs, etc. When it is "bug free," you can deploy it in production, starting it from the Windows service.
- You can run a broad range of programs using this service, including .NET programs, VB scripts and batch files. This service has been used by the open source world to run numerous Java programs.

A little story

My desktop PC in the office was recently re-imaged by the support folks of my company. The first time I signed in, I noticed that a Java program was busy doing something in the background. When I looked deeper, it became clear that this Java program was started by a modified version of the Windows service explained in this article. Later, I found out that another development team in my company got this "great open source tool" from the internet and they modified it to run their Java programs :-).

XYNTService

XYNTService.exe is the name of the executable for this Windows service program. You may get this

Windows service from other websites, but Code Project is the official place to get the most recent version. You can freely use and modify the source code included with this article. I am aware that there are other utility programs providing almost the same functionality as XYNTService. However, as you will see, XYNTService has more features and is a lot easier to use. For example, no editing of the registry is required. Here is how to use the program:

- To install the service, run the following at the command prompt: `XYNTService -i`.
- To un-install the service, run the following at the command prompt: `XYNTService -u`.

By default, the installed service will be started automatically when you reboot the computer. You can also start and shut down the service from the Control Panel or the Administrative Tools using the Services option. When the service is started, it will create all of the processes you defined in the *XYNTService.ini* file one by one. When the service is shut down, it will terminate each of the processes it created in reverse order. The *XYNTService.ini* file should be placed in the same directory as the executable. Here is a sample of the file:

```
[Settings]
ServiceName = XYNTService
CheckProcessSeconds = 30
[Process0]
CommandLine = c:\winnt\system32\notepad.exe
WorkingDir= c:\
PauseStart= 1000
PauseEnd= 1000
UserInterface = Yes
Restart = Yes
[Process1]
CommandLine = java.exe MyPackage.MyClass
Restart = No
UserName =
Domain =
Password =
```

The `ServiceName` property specifies the name you want to use for this NT service. The default name is `XYNTService`. If you copy the executable and the INI file into a different directory and then modify the `ServiceName` property in the INI file, you can install and configure a different service!

The sections `[Process0]`, `[Process1]`, ..., etc. define the properties related to each of these processes. As you can see, there are two processes to create in this example. *notepad.exe* and *java.exe* are the names of these programs. You can specify the parameters for each of these processes in the `CommandLine` property. You **must** specify the full path of the executable file for the corresponding process in the `CommandLine` property unless the executable is already in the system path.

The `CheckProcessSeconds` property specifies if and how often you want to check the processes started by XYNTService. If the property has value 0, then no checking is done. If, for example, the property value is 30 then every 30 seconds XYNTService will query the operating system to see if the processes it started are still running. The dead ones will be restarted if the `Restart` property value (explained later) is defined as `Yes` for that process. The default value of this property, if you don't specify it, is 600 (10 minutes).

Note: In the previous versions of XYNTService, the `ProcCount` value specified how many processes were started by XYNTService. This is no longer required. The `CheckProcess` value is the number of minutes instead of seconds. The current version will work as before if you use `CheckProcess` instead of `CheckProcessSeconds`; you should not use both.

The `WorkingDir` property is the working directory of the current process. If you don't specify this property, then the working directory of the current process will be c:\winnt\system32. The `PauseStart` property is the number of milliseconds the service will wait after starting the current process and before starting the next process. This is useful where the next process depends on the previous process. For example, the second process has to "connect" to the first process in such a way that it does not run until the first process is finished with initialization. If you don't specify the `PauseStart` property, the default value will be 100 milliseconds.

When XYNTService is shut down, it will post `WM_QUIT` messages to the processes it created first and then call the Win32 function `TerminateProcess`. The `PauseEnd` property is the number of milliseconds the service will wait before `TerminateProcess` is called. This property can be used to give a process (started by XYNTService) a chance to clean up and shutdown itself. If you don't specify the `PauseEnd` property, the default value will be 100 milliseconds.

The `UserInterface` property controls whether a logged-in user can see the processes created by XYNTService. However, this only works when XYNTService is running under the local system account, which is the default. In this case, processes created by XYNTService will not be able to access a specific user's settings, such as mapped network drives, etc. You can configure XYNTService to run under a user account, which is done easily from the Control Panel or Administrative Tools. Just select Services and then double click XYNTService in the installed services list to bring up a dialog box.

The `Restart` property is used to decide whether you want XYNTService to restart a dead process. If this property is `No`, which is the default if you don't specify any value, then the corresponding process will not be restarted. If this property is `Yes`, then the dead process will be restarted by XYNTService. See the `CheckProcess` property above on how often dead processes are restarted. You can **bounce** (stop and restart) any process defined in the INI file from the command line. For example, the following command...

```
XYNTService -b 2
```

...will stop and restart the process defined in the `[Process2]` section of the INI file. XYNTService can also be used to start and stop other services from the command line. Here are the commands to start (run) and stop (kill) other services:

```
XYNTService -r NameOfServiceToRun
XYNTService -k NameOfServiceToKill
```

In particular, you can use the above commands to start and stop XYNTService from the command line! **Important note**: You cannot start XYNTService by running it from the command prompt without an argument.

All errors while running XYNTService are written into a log file in the same directory as the executable. The error code in the log file is a decimal number returned by the `GetLastError` API. You can look for it on MSDN.

Impersonating a user (new feature)

The UserName, Domain, and Password properties are used to impersonate a user. If UserName and Password are not empty, then the corresponding process will be started using the specified user account. If Domain is empty, then the specified user account must be an account on the local machine instead of a network account. In order to impersonate a user, the service must be run by a local system. Also, the corresponding user account must have the "logon as service" privilege. If you configure a service to run under a user account, that account will acquire the "logon as service" privilege. Please note that you won't be able to see the user interface of an impersonated process.

Since we may need to put a user password in the INI file, it is important to make sure that the INI file is only accessible by administrators and the local system. The following command will protect the current folder, making it accessible only by administrators and the local system.

```
cacls . /g system:F administrators:F
```

Frequently asked questions

- **Why can't XYNTService start my program**? There could be many reasons. The likely most ones are that you did not give the correct path of the executable in the *XYNTService.ini* file or that your program is located on a mapped network drive (see the following question).
- **My program works fine outside of XYNTService; why does it fail when started by XYNTService**? XYNTService is running under the "local system" account by default. Any

program started by it will also use this account. Your program may need some resources that are not available to this account. For example, "local system" cannot access the current user's registry settings, nor can it access a mapped network drive or any other resource on the LAN. However, you can change the account used by XYNTService. This topic is covered in the question below.

- **How do I change the account that XYNTService uses**? On Windows 2000 or XP, use the Adminstrative Tools menu, select Services and then double click XYNTService from the displayed list to change the logon information (domain name, user name and password). On Windows NT 4.0, use the Control Panel, select/open the Services icon and then double click XYNTService from the displayed list to change the logon information. Please note that you cannot see your program started by XYNTService if XYNTService is not using the "local system" account.
- **How do I turn my program into a Windows service**? You can't, unless you rewrite your program from scratch. A service is a special program that requires some special knowledge to write. If you don't want to learn how to write a service, then use XYNTService to start your program as described in this article so that your program behaves like a service.
- **How do I debug a service**? First, build your service program in debug mode from Visual Studio and set break points at the appropriate places. Then install and start the service. Attach the debugger to the service executable, which should already be running. Note that a service has to be started before the process can be attached to the debugger, so some parts of it can never be stepped into from the debugger. In fact, the most useful code in XYNTService cannot be debugged because the code is executed while XYNTService is started.
- **Why won't my Java program run under XYNTService or why does it die when the user logs off the machine**? I think this could be due to a bug in Java itself. I had some problems with recent versions of JDK, but not with JDK 1.2.2. If you cannot use JDK 1.2.2, then try the *Xrs* option when using Java. Also, change the account used by XYNTService to a LAN user instead of the default "local system" account. This may help you to get around the problem.

History

- 22nd Feb, 08
  - Download recompiled in VS2005 to stop bogus anti-virus warnings.
- 6th Jun, 07
  - Updated source files and article.
- 12th Jan, 07
  - Updated article.
- 24th Feb, 06
  - Updated source code and article.
- 28th Feb, 04
  - Updated source files.
- 7th Sep, 00
  - Original version posted.

License

About the Author

Xiangyang Liu 刘向阳

Location: United States

Member

Discussions and Feedback

1313 messages have been posted for this article. Visit **http://www.codeproject.com/KB/system/xyntservice.aspx** to post and view comments on this article, or click **here** to get a print view with messages.