Installutil.exe (Installer Tool) - .NET Framework

By gewarren

Archived: 2026-04-06 00:43:16 UTC

The Installer tool is a command-line utility that allows you to install and uninstall server resources by executing the installer components in specified assemblies. This tool works in conjunction with classes in the System.Configuration.Install namespace.

This tool is automatically installed with Visual Studio. To run the tool, use Visual Studio Developer Command Prompt or Visual Studio Developer PowerShell.

At the command prompt, type the following:

```
installutil [/u[ninstall]] [options] assembly [[options] assembly] ...
```

Argument	Description
`assembly`	The file name of the assembly in which to execute the installer components. Omit this parameter if you want to specify the assembly's strong name by using the `/AssemblyName` option.

Option	Description

`/h[elp]`
-or-
`/?`
	Displays command syntax and options for the tool.

`/help` *assembly* -or- `/?` *assembly*	Displays additional options recognized by individual installers within the specified assembly, along with command syntax and options for InstallUtil.exe. This option adds the text returned by each installer component's Installer.HelpText property to the help text of InstallUtil.exe. For example, if ServiceProcessInstaller.Account is `User`, the `/username` and `/password` options are available.

Option	Description
`/AssemblyName` "*assemblyName* ,Version=*major.minor.build.revision* ,Culture=*locale* ,PublicKeyToken=*publicKeyToken*"	Specifies the strong name of an assembly, which must be registered in the global assembly cache. The assembly name must be fully qualified with the version, culture, and public key token of the assembly. The fully qualified name must be surrounded by quotes. For example, "myAssembly, Culture=neutral, PublicKeyToken=0038abc9deabfle5, Version=4.0.0.0" is a fully qualified assembly name.
`/InstallStateDir=[` *directoryName* `]`	Specifies the directory of the .InstallState file that contains the data used to uninstall the assembly. The default is the directory that contains the assembly.
`/LogFile=` [*filename*]	Specifies the name of the log file where installation progress is recorded. By default, if the `/LogFile` option is omitted, a log file named *assemblyname*.InstallLog is created. If *filename* is omitted, no log file is generated.
`/LogToConsole` ={ `true` | `false` }	If `true`, displays output to the console. If `false` (the default), suppresses output to the console.
`/ShowCallStack`	Outputs the call stack to the log file if an exception occurs at any point during installation.
`/u` [`ninstall`]	Uninstalls the specified assemblies. Unlike the other options, `/u` applies to all assemblies regardless of where the option appears on the command line.

Individual installers used within an assembly may recognize options in addition to those listed in the Options section. To learn about these options, run InstallUtil.exe with the paths of the assemblies on the command line along with the `/?` or `/help` option. To specify these options, you include them on the command line along with the options recognized by InstallUtil.exe.

Note

Help text on the options supported by individual installer components is returned by the Installer.HelpText property. The individual options that have been entered on the command line are accessible programmatically from the Installer.Context property.

All options and command-line parameters are written to the installation log file. However, if you use the `/Password` parameter, which is recognized by some installer components, the password information is replaced by eight asterisks (*) and won't appear in the log file.

Important

In some cases, parameters passed to the installer may include sensitive or personally identifiable information, which, by default, is written to a plain text log file. To prevent this behavior, you can suppress the log file by specifying `/LogFile=` (with no *filename* argument) on the command line.

.NET Framework applications consist of traditional program files and associated resources, such as message queues, event logs, and performance counters, that must be created when the application is deployed. You can use an assembly's installer components to create these resources when your application is installed and to remove them when your application is uninstalled. Installutil.exe detects and executes these installer components.

You can specify multiple assemblies on the same command line. Any option that occurs before an assembly name applies to that assembly's installation. Except for `/u` and `/AssemblyName`, options are cumulative but overridable. That is, options specified for one assembly apply to all subsequent assemblies unless the option is specified with a new value.

If you run Installutil.exe against an assembly without specifying any options, it places the following three files into the assembly's directory:

- InstallUtil.InstallLog - Contains a general description of the installation progress.
- *assemblyname*.InstallLog - Contains information specific to the commit phase of the installation process. For more information about the commit phase, see the Commit method.
- *assemblyname*.InstallState - Contains data used to uninstall the assembly.

Installutil.exe uses reflection to inspect the specified assemblies and to find all Installer types that have the System.ComponentModel.RunInstallerAttribute attribute set to `true`. The tool then executes either the Installer.Install or the Installer.Uninstall method on each instance of the Installer type. Installutil.exe performs installation in a transactional manner; that is, if one of the assemblies fails to install, it rolls back the installations of all other assemblies. Uninstall is not transactional.

Installutil.exe cannot install or uninstall delay-signed assemblies, but it can install or uninstall strong-named assemblies.

The 32-bit version of the common language runtime (CLR) ships with only the 32-bit version of the Installer tool, but the 64-bit version of the CLR ships with both 32-bit and 64-bit versions of the Installer tool. When using the 64-bit CLR, use the 32-bit Installer tool to install 32-bit assemblies, and the 64-bit Installer tool to install 64-bit and common intermediate language (CIL) assemblies. Both versions of the Installer tool behave the same.

You can't use Installutil.exe to deploy a Windows service that was created by using C++, because Installutil.exe doesn't recognize the embedded native code that's produced by the C++ compiler. If you try to deploy a C++ Windows service with Installutil.exe, an exception such as BadImageFormatException will be thrown. To work with this scenario, move the service code to a C++ module, and then write the installer object in C# or Visual Basic.

The following command displays a description of the command syntax and options for InstallUtil.exe.

```
installutil /?
```

The following command displays a description of the command syntax and options for InstallUtil.exe. It also displays a description and list of options supported by the installer components in `myAssembly.exe` if help text has been assigned to the installer's Installer.HelpText property.

```
installutil /? myAssembly.exe
```

The following command executes the installer components in the assembly `myAssembly.exe`.

```
installutil myAssembly.exe
```

The following command executes the installer components in an assembly by using the `/AssemblyName` switch and a fully qualified name.

```
installutil /AssemblyName "myAssembly, Culture=neutral, PublicKeyToken=0038abc9deabfle5, Version=4.0.0.0"
```

The following command executes the installer components in an assembly specified by file name and in an assembly specified by strong name. Note that all assemblies specified by file name must precede assemblies specified by strong name on the command line, because the `/AssemblyName` option cannot be overridden.

```
installutil myAssembly.exe /AssemblyName "myAssembly, Culture=neutral, PublicKeyToken=0038abc9deabfle5, Versio
```

The following command executes the uninstaller components in the assembly `myAssembly.exe`.

```
installutil /u myAssembly.exe
```

The following command executes the uninstaller components in the assemblies `myAssembly1.exe` and `myAssembly2.exe`.

```
installutil myAssembly1.exe /u myAssembly2.exe
```

Because the position of the `/u` option on the command line is not important, this is equivalent to the following command.

```
installutil /u myAssembly1.exe myAssembly2.exe
```

The following command executes the installers in the assembly `myAssembly.exe` and specifies that progress information will be written to `myLog.InstallLog`.

```
installutil /LogFile=myLog.InstallLog myAssembly.exe
```

The following command executes the installers in the assembly `myAssembly.exe`, specifies that progress information should be written to `myLog.InstallLog`, and uses the installers' custom `/reg` option to specify that updates should be made to the system registry.

```
installutil /LogFile=myLog.InstallLog /reg=true myAssembly.exe
```

The following command executes the installers in the assembly `myAssembly.exe`, uses the installer's custom `/email` option to specify the user's email address, and suppresses output to the log file.

```
installutil /LogFile= /email=admin@mycompany.com myAssembly.exe
```

The following command writes the installation progress for `myAssembly.exe` to `myLog.InstallLog` and writes the progress for `myTestAssembly.exe` to `myTestLog.InstallLog`.

```
installutil /LogFile=myLog.InstallLog myAssembly.exe /LogFile=myTestLog.InstallLog myTestAssembly.exe
```

- System.Configuration.Install
- Tools
- Developer command-line shells
