Restarting Windows from within a .NET application

Asked 15 years, 11 months ago Modified 6 years, 6 months ago

Viewed 25k times

23

How could I restart or shutdown Windows using the .NET framework?

c# .net winforms

Share

Improve this question

Follow

edited Apr 10, 2012 at 16:57

abatishchev

100k ● 88 ● 301 ● 442

asked Jan 20, 2009 at 18:03

Terenced

675 ● 2 ● 6 ● 17

2 The answers provided here will most probably be of a higher quality. If you don't know the best solution to a problem yet, how could you know that the first hit on Google is the best way to solve it? Btw, the first hit for me was a dead link...
– Dirk Vollmar Jan 20, 2009 at 18:13

2 I'd guess that 80% of the programming-related questions here can easily be answered by using Google, but the purpose of this site is also to be a knowledge base providing high-quality answers which will one day become the first hit on Google. – Dirk Vollmar Jan 20, 2009 at 18:27

8 Answers

Sorted by: Highest score (default)

46

The following code will execute the shutdown command from the shell:

```
// using System.Diagnostics;

class Shutdown
{
    /// <summary>
    /// Windows restart
    /// </summary>
    public static void Restart()
    {
        StartShutDown("-f -r -t 5");
    }

    /// <summary>
    /// Log off.
    /// </summary>
    public static void LogOff()
    {
        StartShutDown("-l");
    }

    /// <summary>
    ///  Shutting Down Windows
    /// </summary>
    public static void Shut()
    {
        StartShutDown("-f -s -t 5");
    }
```

```
    private static void StartShutDown(string param)
    {
        ProcessStartInfo proc = new ProcessStartInfo()
        proc.FileName = "cmd";
        proc.WindowStyle = ProcessWindowStyle.Hidden;
        proc.Arguments = "/C shutdown " + param;
        Process.Start(proc);
    }
}
```

(Source: http://dotnet-snippets.de/dns/c-windows-herrunterfahren-ausloggen-neustarten-SID455.aspx

Share Improve this answer

Follow

answered Jan 20, 2009 at 18:07

Dirk Vollmar

176k ● 53 ● 259 ● 316

1 +1 Simple solution for what could have been a complex question. – dance2die Apr 4, 2009 at 18:12

-f will be auto with -t > 0 so can be skiped – user1005462 Sep 16, 2023 at 9:21

8

Here is P/Invoke way:

```
namespace pcController
{
    /// <summary>
    /// Specifies the type of restart options that an
    /// </summary>
    public enum RestartOptions
    {
        /// <summary>
        /// Shuts down all processes running in the se
process that called the ExitWindowsEx function. Then i
```

```
        /// </summary>
        LogOff = 0,
        /// <summary>
        /// Shuts down the system and turns off the po
support the power-off feature.
        /// </summary>
        PowerOff = 8,
        /// <summary>
        /// Shuts down the system and then restarts th
        /// </summary>
        Reboot = 2,
        /// <summary>
        /// Shuts down the system to a point at which
the power. All file buffers have been flushed to disk,
processes have stopped. If the system supports the pow
is also turned off.
        /// </summary>
        ShutDown = 1,
        /// <summary>
        /// Suspends the system.
        /// </summary>
        Suspend = -1,
        /// <summary>
        /// Hibernates the system.
        /// </summary>
        Hibernate = -2,
    }
    /// <summary>
    /// An LUID is a 64-bit value guaranteed to be uni
which it was generated. The uniqueness of a locally un
guaranteed only until the system is restarted.
    /// </summary>
    [StructLayout(LayoutKind.Sequential, Pack = 1)]
    internal struct LUID
    {
        /// <summary>
        /// The low order part of the 64 bit value.
        /// </summary>
        public int LowPart;
        /// <summary>
        /// The high order part of the 64 bit value.
        /// </summary>
        public int HighPart;
    }
```

```
    /// <summary>
    /// The LUID_AND_ATTRIBUTES structure represents a
identifier (LUID) and its attributes.
    /// </summary>
    [StructLayout(LayoutKind.Sequential, Pack = 1)]
    internal struct LUID_AND_ATTRIBUTES
    {
        /// <summary>
        /// Specifies an LUID value.
        /// </summary>
        public LUID pLuid;
        /// <summary>
        /// Specifies attributes of the LUID. This val
bit flags. Its meaning is dependent on the definition
        /// </summary>
        public int Attributes;
    }
    /// <summary>
    /// The TOKEN_PRIVILEGES structure contains inform
privileges for an access token.
    /// </summary>
    [StructLayout(LayoutKind.Sequential, Pack = 1)]
    internal struct TOKEN_PRIVILEGES
    {
        /// <summary>
        /// Specifies the number of entries in the Pri
        /// </summary>
        public int PrivilegeCount;
        /// <summary>
        /// Specifies an array of LUID_AND_ATTRIBUTES
structure contains the LUID and attributes of a privil
        /// </summary>
        public LUID_AND_ATTRIBUTES Privileges;
    }
    /// <summary>
    /// Implements methods to exit Windows.
    /// </summary>
    public class LibClass
    {
        /// <summary>Required to enable or disable the
token.</summary>
        private const int TOKEN_ADJUST_PRIVILEGES = 0x
        /// <summary>Required to query an access token
        private const int TOKEN_QUERY = 0x8;
```

```
        /// <summary>The privilege is enabled.</summar
        private const int SE_PRIVILEGE_ENABLED = 0x2;
        /// <summary>Specifies that the function shoul
message-table resource(s) for the requested message.</
        private const int FORMAT_MESSAGE_FROM_SYSTEM =
        /// <summary>Forces processes to terminate. Wh
system does not send the WM_QUERYENDSESSION and WM_END
can cause the applications to lose data. Therefore, yo
flag in an emergency.</summary>
        private const int EWX_FORCE = 4;
        /// <summary>
        /// The LoadLibrary function maps the specifie
the address space of the calling process.
        /// </summary>
        /// <param name="lpLibFileName">Pointer to a n
that names the executable module (either a .dll or .ex
specified is the file name of the module and is not re
in the library module itself, as specified by the LIBR
module-definition (.def) file.</param>
        /// <returns>If the function succeeds, the ret
the module.<br></br><br>If the function fails, the ret
extended error information, call Marshal.GetLastWin32E
        [DllImport("kernel32.dll", EntryPoint = "LoadL
CharSet.Ansi)]
        private static extern IntPtr LoadLibrary(strin
        /// <summary>
        /// The FreeLibrary function decrements the re
loaded dynamic-link library (DLL). When the reference
module is unmapped from the address space of the calli
is no longer valid.
        /// </summary>
        /// <param name="hLibModule">Handle to the loa
LoadLibrary or GetModuleHandle function returns this h
        /// <returns>If the function succeeds, the ret
</br><br>If the function fails, the return value is ze
information, call Marshal.GetLastWin32Error.</br></ret
        [DllImport("kernel32.dll", EntryPoint = "FreeL
CharSet.Ansi)]
        private static extern int FreeLibrary(IntPtr h
        /// <summary>
        /// The GetProcAddress function retrieves the
function or variable from the specified dynamic-link l
        /// </summary>
        /// <param name="hModule">Handle to the DLL mo
```

```
function or variable. The LoadLibrary or GetModuleHand
handle.</param>
        /// <param name="lpProcName">Pointer to a null
containing the function or variable name, or the funct
this parameter is an ordinal value, it must be in the
order word must be zero.</param>
        /// <returns>If the function succeeds, the ret
of the exported function or variable.<br></br><br>If t
return value is NULL. To get extended error informatio
Marshal.GetLastWin32Error.</br></returns>
        [DllImport("kernel32.dll", EntryPoint = "GetPr
CharSet.Ansi)]
        private static extern IntPtr GetProcAddress(In
lpProcName);
        /// <summary>
        /// The SetSuspendState function suspends the
down. Depending on the Hibernate parameter, the system
(sleep) state or hibernation (S4). If the ForceFlag pa
system suspends operation immediately; if it is FALSE,
permission from all applications and device drivers be
        /// </summary>
        /// <param name="Hibernate">Specifies the stat
the system hibernates. If FALSE, the system is suspend
        /// <param name="ForceCritical">Forced suspens
function broadcasts a PBT_APMSUSPEND event to each app
immediately suspends operation. If FALSE, the function
PBT_APMQUERYSUSPEND event to each application to reque
operation.</param>
        /// <param name="DisableWakeEvent">If TRUE, th
wake events. If FALSE, any system wake events remain e
        /// <returns>If the function succeeds, the ret
</br><br>If the function fails, the return value is ze
information, call Marshal.GetLastWin32Error.</br></ret
        [DllImport("powrprof.dll", EntryPoint = "SetSu
CharSet.Ansi)]
        private static extern int SetSuspendState(int
ForceCritical, int DisableWakeEvent);
        /// <summary>
        /// The OpenProcessToken function opens the ac
with a process.
        /// </summary>
        /// <param name="ProcessHandle">Handle to the
token is opened.</param>
        /// <param name="DesiredAccess">Specifies an a
```

```
the requested types of access to the access token. The
are compared with the token's discretionary access-con
determine which accesses are granted or denied.</param
        /// <param name="TokenHandle">Pointer to a han
newly-opened access token when the function returns.</
        /// <returns>If the function succeeds, the ret
</br><br>If the function fails, the return value is ze
information, call Marshal.GetLastWin32Error.</br></ret
        [DllImport("advapi32.dll", EntryPoint = "OpenP
CharSet.Ansi)]
        private static extern int OpenProcessToken(Int
DesiredAccess, ref IntPtr TokenHandle);
        /// <summary>
        /// The LookupPrivilegeValue function retrieve
identifier (LUID) used on a specified system to locall
privilege name.
        /// </summary>
        /// <param name="lpSystemName">Pointer to a nu
specifying the name of the system on which the privile
a null string is specified, the function attempts to f
the local system.</param>
        /// <param name="lpName">Pointer to a null-ter
specifies the name of the privilege, as defined in the
example, this parameter could specify the constant SE_
corresponding string, "SeSecurityPrivilege".</param>
        /// <param name="lpLuid">Pointer to a variable
locally unique identifier by which the privilege is kn
specified by the lpSystemName parameter.</param>
        /// <returns>If the function succeeds, the ret
</br><br>If the function fails, the return value is ze
information, call Marshal.GetLastWin32Error.</br></ret
        [DllImport("advapi32.dll", EntryPoint = "Looku
CharSet = CharSet.Ansi)]
        private static extern int LookupPrivilegeValue
string lpName, ref LUID lpLuid);
        /// <summary>
        /// The AdjustTokenPrivileges function enables
in the specified access token. Enabling or disabling p
token requires TOKEN_ADJUST_PRIVILEGES access.
        /// </summary>
        /// <param name="TokenHandle">Handle to the ac
the privileges to be modified. The handle must have TO
access to the token. If the PreviousState parameter is
must also have TOKEN_QUERY access.</param>
```

```
        /// <param name="DisableAllPrivileges">Specifi
disables all of the token's privileges. If this value
disables all privileges and ignores the NewState param
function modifies privileges based on the information
NewState parameter.</param>
        /// <param name="NewState">Pointer to a TOKEN_
specifies an array of privileges and their attributes.
DisableAllPrivileges parameter is FALSE, AdjustTokenPr
disables these privileges for the token. If you set th
attribute for a privilege, the function enables that p
disables the privilege. If DisableAllPrivileges is TRU
this parameter.</param>
        /// <param name="BufferLength">Specifies the s
buffer pointed to by the PreviousState parameter. This
the PreviousState parameter is NULL.</param>
        /// <param name="PreviousState">Pointer to a b
fills with a TOKEN_PRIVILEGES structure that contains
privileges that the function modifies. This parameter
        /// <param name="ReturnLength">Pointer to a va
required size, in bytes, of the buffer pointed to by t
parameter. This parameter can be NULL if PreviousState
        /// <returns>If the function succeeds, the ret
determine whether the function adjusted all of the spe
Marshal.GetLastWin32Error.</returns>
        [DllImport("advapi32.dll", EntryPoint = "Adjus
CharSet = CharSet.Ansi)]
        private static extern int AdjustTokenPrivilege
DisableAllPrivileges, ref TOKEN_PRIVILEGES NewState, i
TOKEN_PRIVILEGES PreviousState, ref int ReturnLength);
        /// <summary>
        /// The ExitWindowsEx function either logs off
down the system, or shuts down and restarts the system
WM_QUERYENDSESSION message to all applications to dete
terminated.
        /// </summary>
        /// <param name="uFlags">Specifies the type of
        /// <param name="dwReserved">This parameter is
        /// <returns>If the function succeeds, the ret
</br><br>If the function fails, the return value is ze
information, call Marshal.GetLastWin32Error.</br></ret
        [DllImport("user32.dll", EntryPoint = "ExitWin
CharSet.Ansi)]
        private static extern int ExitWindowsEx(int uF
        /// <summary>
```

```
        /// The FormatMessage function formats a messa
requires a message definition as input. The message de
buffer passed into the function. It can come from a me
an already-loaded module. Or the caller can ask the fu
system's message table resource(s) for the message def
finds the message definition in a message table resour
identifier and a language identifier. The function cop
text to an output buffer, processing any embedded inse
requested.
        /// </summary>
        /// <param name="dwFlags">Specifies aspects of
and how to interpret the lpSource parameter. The low-o
specifies how the function handles line breaks in the
order byte can also specify the maximum width of a for
</param>
        /// <param name="lpSource">Specifies the locat
definition. The type of this parameter depends upon th
parameter.</param>
        /// <param name="dwMessageId">Specifies the me
requested message. This parameter is ignored if dwFlag
FORMAT_MESSAGE_FROM_STRING.</param>
        /// <param name="dwLanguageId">Specifies the l
the requested message. This parameter is ignored if dw
FORMAT_MESSAGE_FROM_STRING.</param>
        /// <param name="lpBuffer">Pointer to a buffer
null-terminated) message. If dwFlags includes FORMAT_M
the function allocates a buffer using the LocalAlloc f
pointer to the buffer at the address specified in lpBu
        /// <param name="nSize">If the FORMAT_MESSAGE_
not set, this parameter specifies the maximum number o
stored in the output buffer. If FORMAT_MESSAGE_ALLOCAT
parameter specifies the minimum number of TCHARs to al
buffer. For ANSI text, this is the number of bytes; fo
the number of characters.</param>
        /// <param name="Arguments">Pointer to an arra
as insert values in the formatted message. A %1 in the
the first value in the Arguments array; a %2 indicates
so on.</param>
        /// <returns>If the function succeeds, the ret
of TCHARs stored in the output buffer, excluding the t
character.<br></br><br>If the function fails, the retu
extended error information, call Marshal.GetLastWin32E
        [DllImport("user32.dll", EntryPoint = "FormatM
CharSet.Ansi)]
```

```
        private static extern int FormatMessage(int dw
int dwMessageId, int dwLanguageId, StringBuilder lpBuf
Arguments);
        /// <summary>
        /// Exits windows (and tries to enable any req
necesarry).
        /// </summary>
        /// <param name="how">One of the RestartOption
how to exit windows.</param>
        /// <param name="force">True if the exit has t
otherwise.</param>
        /// <exception cref="PrivilegeException">There
requesting a required privilege.</exception>
        /// <exception cref="PlatformNotSupportedExcep
method is not supported on this platform.</exception>
        public static void ExitWindows(RestartOptions
        {
            switch (how)
            {
                case RestartOptions.Suspend:
                    SuspendSystem(false, force);
                    break;
                case RestartOptions.Hibernate:
                    SuspendSystem(true, force);
                    break;
                default:
                    ExitWindows((int)how, force);
                    break;
            }
        }
        /// <summary>
        /// Exits windows (and tries to enable any req
necesarry).
        /// </summary>
        /// <param name="how">One of the RestartOption
how to exit windows.</param>
        /// <param name="force">True if the exit has t
otherwise.</param>
        /// <remarks>This method cannot hibernate or s
</remarks>
        /// <exception cref="PrivilegeException">There
requesting a required privilege.</exception>
        protected static void ExitWindows(int how, boo
        {
```

```
            EnableToken("SeShutdownPrivilege");
            if (force)
                how = how | EWX_FORCE;
            if (ExitWindowsEx(how, 0) == 0)
                throw new
PrivilegeException(FormatError(Marshal.GetLastWin32Err
        }
        /// <summary>
        /// Tries to enable the specified privilege.
        /// </summary>
        /// <param name="privilege">The privilege to e
        /// <exception cref="PrivilegeException">There
requesting a required privilege.</exception>
        protected static void EnableToken(string privi
        {
            if (!CheckEntryPoint("advapi32.dll", "Adju
                return;
            IntPtr tokenHandle = IntPtr.Zero;
            LUID privilegeLUID = new LUID();
            TOKEN_PRIVILEGES newPrivileges = new TOKEN
            TOKEN_PRIVILEGES tokenPrivileges;
            if (OpenProcessToken(Process.GetCurrentPro
TOKEN_ADJUST_PRIVILEGES | TOKEN_QUERY, ref tokenHandle
                throw new
PrivilegeException(FormatError(Marshal.GetLastWin32Err
            if (LookupPrivilegeValue("", privilege, re
                throw new
PrivilegeException(FormatError(Marshal.GetLastWin32Err
            tokenPrivileges.PrivilegeCount = 1;
            tokenPrivileges.Privileges.Attributes = SE
            tokenPrivileges.Privileges.pLuid = privile
            int size = 4;
            if (AdjustTokenPrivileges(tokenHandle, 0,
(12 * tokenPrivileges.PrivilegeCount), ref newPrivileg
                throw new
PrivilegeException(FormatError(Marshal.GetLastWin32Err
        }
        /// <summary>
        /// Suspends or hibernates the system.
        /// </summary>
        /// <param name="hibernate">True if the system
if the system has to be suspended.</param>
        /// <param name="force">True if the exit has t
otherwise.</param>
```

```
        /// <exception cref="PlatformNotSupportedExcep
method is not supported on this platform.</exception>
        protected static void SuspendSystem(bool hiber
        {
            if (!CheckEntryPoint("powrprof.dll", "SetS
                throw new PlatformNotSupportedExceptio
method is not supported on this system!");
            SetSuspendState((int)(hibernate ? 1 : 0),
        }
        /// <summary>
        /// Checks whether a specified method exists o
        /// </summary>
        /// <param name="library">The library that hol
        /// <param name="method">The entry point of th
</param>
        /// <returns>True if the specified method is p
</returns>
        protected static bool CheckEntryPoint(string l
        {
            IntPtr libPtr = LoadLibrary(library);
            if (!libPtr.Equals(IntPtr.Zero))
            {
                if (!GetProcAddress(libPtr, method).Eq
                {
                    FreeLibrary(libPtr);
                    return true;
                }
                FreeLibrary(libPtr);
            }
            return false;
        }
        /// <summary>
        /// Formats an error number into an error mess
        /// </summary>
        /// <param name="number">The error number to c
        /// <returns>A string representation of the sp
</returns>
        protected static string FormatError(int number
        {
            StringBuilder buffer = new StringBuilder(2
            FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM,
buffer, buffer.Capacity, 0);
            return buffer.ToString();
        }
```

```
    }
    /// <summary>
    /// The exception that is thrown when an error occ
specific privilege.
    /// </summary>
    public class PrivilegeException : Exception
    {
        /// <summary>
        /// Initializes a new instance of the Privileg
        /// </summary>
        public PrivilegeException() : base() { }
        /// <summary>
        /// Initializes a new instance of the Privileg
specified error message.
        /// </summary>
        /// <param name="message">The message that des
        public PrivilegeException(string message) : ba
    }
}
```

Share Improve this answer Follow

edited Aug 31, 2012 at 7:46

answered Jan 20, 2009 at 18:45

abatishchev
100k ● 88 ● 301 ● 442

While this link may answer the question, it is better to include the essential parts of the answer here and provide the link for reference. Link-only answers can become invalid if the linked page changes. – Himanshu Aug 31, 2012 at 3:34

1 @hims056: Absolutely, this was my very old post :) – abatishchev Aug 31, 2012 at 7:46

Nice. This came in my attention when I was reviewing Low Quality Posts – Himanshu Aug 31, 2012 at 7:48

7

I don't know a pure .NET way to do it. Your options include:

- P/Invoke the ExitWindowsEx Win32 function
- Use Process.Start to run shutdown.exe as already suggested.

Share Improve this answer

Follow

answered Jan 20, 2009 at 18:09

driis

164k ● 46 ● 270 ● 345

I need to do this, and I already found the ExitWindowsEx function. I am using this in my code (C++), but it is not working: `ExitWindowsEx(EWX_REBOOT, 5);`. I read on MSDN that it is better to do this than to use `System("shutdown.exe -r -t 05");` What is wrong with my code? – user2509848 Dec 26, 2013 at 16:18

It might be an access rights problem. Does the user the code is running under, have rights to reboot the machine ? – driis Dec 27, 2013 at 13:54

I do not know. I am in a standard account, but I did try it in an admin account. UAC may have blocked it - I keep it set all the way high. Also, I am on Windows 8.1, which is reputed to have excellent security features. I solved the problem with `system("shutdown -s");`. Thanks for your help. – user2509848 Dec 27, 2013 at 15:14

6

The best way I saw:

```
System.Diagnostics.Process.Start("cmd.exe /c shutdown"
```

Also there are few WinAPI ways..

Share Improve this answer

Follow

answered Jan 20, 2009 at 18:07

abatishchev

100k ● 88 ● 301 ● 442

FWIW, this failed for me, but a variation worked: System.Diagnostics.Process.Start("cmd.exe", $"/c shutdown /r /f /t {afterSeconds}"); Adding the seconds allows your application to shut down and shows a message on the screen. – mj2008 Nov 14, 2019 at 9:55

6

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
// Remember to add a reference to the System.Managemen
using System.Management;

namespace ShutDown
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnShutDown_Click(object sender,
        {
            ManagementBaseObject mboShutdown = null;
            ManagementClass mcWin32 = new
ManagementClass("Win32_OperatingSystem");
            mcWin32.Get();
```

```
            // You can't shutdown without security pri
            mcWin32.Scope.Options.EnablePrivileges = t
            ManagementBaseObject mboShutdownParams =
mcWin32.GetMethodParameters("Win32Shutdown");
            // Flag 1 means we want to shut down the s
            mboShutdownParams["Flags"] = "1";
            mboShutdownParams["Reserved"] = "0";
            foreach (ManagementObject manObj in mcWin3
            {
                mboShutdown = manObj.InvokeMethod("Win
mboShutdownParams, null);
            }
        }
    }
}
```

Share Improve this answer

Follow

answered Jan 20, 2009 at 18:09

mapache

1,361 • 9 • 11

6

You can use WMI, with LINQ to smooth over its rough spots.

```
const int restart = 2;
var management = new ManagementClass("Win32_OperatingS
management.Scope.Options.EnablePrivileges = true;
management.GetInstances().OfType<ManagementObject>
().First().InvokeMethod("Win32Shutdown", new object[]
```

Share Improve this answer

Follow

answered Jun 1, 2018 at 20:41

Edward Brey

41.6k • 21 • 209 • 265

---

check this aticle.. that gives the answers for ur question

0

http://www.codeproject.com/KB/cs/timercomputershutdown.aspx

Share Improve this answer

Follow

answered Jan 20, 2009 at 18:14

0

If you don't mind using the WinAPI, you could call the ExitWindows function, in user32.dll. Here's an article telling you all about it: http://www.eggheadcafe.com/tutorials/aspnet/e5ef4e3e-6f42-4b9b-8834-04366ce32c96/net-lock-logoff-reboot.aspx

Share Improve this answer

Follow

answered Jan 20, 2009 at 18:14

gkrogers

8,336 ● 3 ● 30 ● 36