Selectively disabling UAC for specific programs on Windows Programatically

There are dozens of posts questions/answers on stack and other forums about disabling/bypassing/suppressing UAC. There are solutions as well. But progmatically perhaps not. I could see only one solution Disabling UAC programmatically which I could find useful as I cud not use it in a program anyway.

Can there be a programatic solution for saving user to be prompted everytime he/she runs a program like wamp and they always have to click yes, So it would be better to tell windows that their choice is always yes. I am sure there would be as

I have found **Here** that windows provides this facility in Task Scheduler through GUI so it must be possible through code as well.

Update : I have prepared a solution which is working for me. See my answer.

c# | scheduled-tasks | uac

edited Mar 4 at 9:34

asked Mar 3 at 21:29

Sami
2,110 9 32

Why are you trying to bypass the UAC – Jakob Bowyer Mar 3 at 21:31

@JakobBowyer Because wherever I have deployed my c#,mysql desktop applications (which need wamp running on system), Users get irritated of UAC dialogue when wamp is started. – Sami Mar 3 at 21:37

that sounds like a foolish idea. What if someone exploits your software, you have already done the UAC bypass for them, your now opening up a massive damn security flaw – Jakob Bowyer Mar 3 at 21:41

3 @JakobBowyer you are too aggressive to discuss. I have no security risks in a desktop application running on personal computer. Still you might be right. You can down vote you can discourage others to guide me. Thats your right. But your language and your arguments are not convincing, but just like showing a disgust. I am sorry but thats what I felt. – Sami Mar 3 at 22:04

1 This question has -4, this other one stackoverflow.com/questions/682182/… has 10. Both questions are about disabling the UAC; one of them has real answers while the other one has opinions. It's sad.

stackoverflow.com/faq#etiquette – daniloquio May 24 at 13:27

show **5** more comments

2 Answers

1. Download Microsoft.Win32.TaskScheduler.dll from This Codeplex link
2. Make a c# application (Windows or Console) and add reference to the above dll
3. Add New Item (Application Manifest File) to your project (this application)
4. Change `<requestedExecutionLevel level="asInvoker" uiAccess="false" />` to `<requestedExecutionLevel level="requireAdministrator" uiAccess="false" />`
5. Write following code in your program.cs file

```
using System;
using Microsoft.Win32.TaskScheduler;
class Program
{
   static void Main(string[] args)
   {
      TaskService ts = new TaskService();
      TaskDefinition td = ts.NewTask();
      td.Principal.RunLevel = TaskRunLevel.Highest;
      //td.Triggers.AddNew(TaskTriggerType.YourDesiredSchedule);
```

```
        td.Triggers.AddNew(TaskTriggerType.Logon);
        //td.Actions.Add(new ExecAction("Path Of your Application File", null));
        td.Actions.Add(new ExecAction(@"c:\wamp\wampmanager.exe", null));
        ts.RootFolder.RegisterTaskDefinition("anyNamefortask", td);
    }
}
```

6.Now compile and run your Application(this app)

Now your application (e.g WAMP) will run without prompting any UAC dialog on your desired schedule (every time your log on windows in my case)

Sources

Initiated from : Can you turn off UAC for a single app? and Selectively disabling UAC for specific programs on Windows 7

Basic Idea from : **Make Vista launch UAC restricted programs at startup with Task Scheduler**

Basic Implementation from **Creating Scheduled Tasks**

edited Nov 4 at 9:21

answered Mar 3 at 23:53

Sami
2,110 9 32

add comment

The proper methodology wouldn't be to ignore the User Access Control (UAC) but rather test within those parameters. That way you don't disrupt security, you instead work within it's confines.

By disabling security, you run the risk of exploits. According to Secuna which provide several security test have noticed that small companies, lazy developer applications, and blatantly disregard for security are applications that have been focused on.

Which means your application may become a victim at some point.

The approach I would take, is test within UAC. Ensure the proper permissions exists to carry out your task, that way it isn't constantly running with Elevated Permission. An example may be:

```
class Elevated_Rights
{
    // Token Bool:
    private bool _level = false;

    #region Constructor:
    protected Elevated_Rights()
    {
          // Invoke Method On Creation:
          Elevate();
     }
     #endregion
     public void Elevate()
     {
          // Get Identity:
          WindowsIdentity user = WindowsIdentity.GetCurrent();

          // Set Principal
          WindowsPrincipal role = new WindowsPrincipal(user);

          #region Test Operating System for UAC:
          if (Environment.OSVersion.Platform != PlatformID.Win32NT ||
           {
               // False:
               _level = false;
           }
           #endregion
           else
           {
                #region Test Identity Not Null:
                if (user == null)
                {
                    // False:
```

Something along those lines would allow you to test against the UAC, then perform a task. I'm not quite

sure why you would like to disable the UAC, but that would be my approach.

Hopefully that helps.

answered Mar 3 at 23:17

Greg
2,131 1 7 33

Thanks for the help – Sami Mar 3 at 23:54

add comment

Not the answer you're looking for? Browse other questions tagged c# scheduled-tasks uac **or ask your own question.**