

Crank Up an App's Response Time

Alter the priority of processes to speed response time for an application, and use the ErrorProvider to get error information about a control.

by Karl E. Peterson and Juval Löwy

Technology Toolbox

- VB.NET
- C#
- SQL Server 2000
- ASP.NET
- XML
- VB6
- Note:

Karl E. Peterson's solution also works with VB5.

Q: Preempt Other Processes

I'm writing an application that must monitor the status of other applications, somewhat similar to the system Task List utility. My application needs to be responsive to the user at all times, but I find that it stalls when another process is performing an intense operation. How can I crank up my app's response time, even when the system is under heavy load?

A:

Most processes running on 32-bit versions of Windows are created equal and receive essentially the same "normal" priority on the CPU's time as any other. Some processes might need to override the default priority assigned to them to preempt other processes, or to run only when

other processes are inactive. In your case, you want to assign a high priority to your process to claim more opportunities for CPU cycles than the system would allocate otherwise. *Use this setting with extreme caution, as a high-priority process has the capability of consuming nearly all available CPU time.*

The procedure for altering a process's priority is relatively straightforward. You first call `OpenProcess` to obtain a handle to the desired process. You must have `PROCESS_SET_INFORMATION` access rights to ensure success. After opening the process, call the `SetPriorityClass` API to set the desired priority level. Clean up by calling `CloseHandle` on the process handle. Follow the same general procedure to read a process's priority setting, but call the `GetPriorityClass` API instead. If you call the

Go Online!

Use these Locator+ codes at www.visualstudiomagazine.com to go directly to these related resources.

Download

VS0211QA Download the code for this article, which includes a drop-in module containing code to set/retrieve a process's priority; and the `ErrorProviderDemo`, a Windows Forms application that shows you how to use the `ErrorProvider` controls.

Discuss

VS0211QA_D Discuss this article in the "Classic" VB forum.

Read More

VS0211QA_T Read this article online.

VB9902AP_T Ask the VB Pro, "Force Your Way to the Foreground," by Karl E. Peterson

NUEP021204AN "Boost Web Power With ASP.NET" by Rob Howard

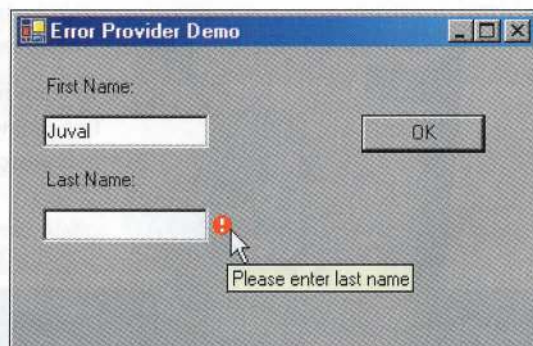


Figure 1. Provide Error Information. The `ErrorProvider` control displays an icon next to the offending control. The icon blinks and displays a tooltip indicating the error's nature to the user.

GetPriorityClass API, you need only PROCESS_QUERY_INFORMATION access rights to the process.

Make these priority adjustments as easy as possible by wrapping them into a pair of routines that accept either a process ID or a window handle as a pointer to the desired process (see Listing 1). Most often, determining your process's handle—required to open the process—involves an extra API call, while you always know the window handle for your forms. So I coded the call to GetWindowThreadProcessId directly within my utility routines. GetWindowThreadProcessId accepts a window handle in its first parameter, and returns the

associated process ID in its second parameter. Call SetProcessPriority by passing a known window handle and the desired priority:

```
Call SetProcessPriority( _
    hWnd:=Me.hWnd, Priority:=ppHigh)
```

You must also be aware that while you're executing within the Visual Basic IDE, you're a part of VB's process. That is, changing "your" application's setting actually changes the setting for VB itself. This might not be desirable. Two approaches you might want

VB5, VB6 • Adjust Your Response

```
Option Explicit

' Win32 API declarations
Private Declare Function _
    GetWindowThreadProcessId Lib "user32" ( _
        ByVal hWnd As Long, lpdwProcessID As Long) _
        As Long
Private Declare Function OpenProcess Lib _
    "kernel32" (ByVal dwDesiredAccess As Long, _
        ByVal bInheritHandle As Long, _
        ByVal dwProcessID As Long) As Long
Private Declare Function SetPriorityClass Lib _
    "kernel32" (ByVal hProcess As Long, _
        ByVal dwPriorityClass As Long) As Long
Private Declare Function GetPriorityClass Lib _
    "kernel32" (ByVal hProcess As Long) As Long
Private Declare Function CloseHandle Lib _
    "kernel32" (ByVal hObject As Long) As Long

' Used by the OpenProcess API call
Private Const PROCESS_QUERY_INFORMATION _
    As Long = &H400
Private Const PROCESS_SET_INFORMATION _
    As Long = &H200

' Used by SetPriorityClass
Private Const NORMAL_PRIORITY_CLASS = &H20
Private Const IDLE_PRIORITY_CLASS = &H40
Private Const HIGH_PRIORITY_CLASS = &H80
Private Const REALTIME_PRIORITY_CLASS = &H100

Public Enum ProcessPriorities
    ppIdle = IDLE_PRIORITY_CLASS
    ppNormal = NORMAL_PRIORITY_CLASS
    ppHigh = HIGH_PRIORITY_CLASS
    ppRealtime = REALTIME_PRIORITY_CLASS
End Enum

Public Function GetProcessPriority(Optional _
    ByVal ProcessID As Long, Optional ByVal hWnd _
    As Long) As Long
    Dim hProc As Long
    Const fdwAccess As Long = _
        PROCESS_QUERY_INFORMATION

    ' If not passed a PID, then find value from hWnd.
    If ProcessID = 0 Then
        Call GetWindowThreadProcessId(hWnd, _
            ProcessID)

        ' Need to open process with simple query
        ' rights, get the current setting, and close
        ' handle.
        hProc = OpenProcess(fdwAccess, 0&, ProcessID)
        GetProcessPriority = GetPriorityClass(hProc)
        Call CloseHandle(hProc)
    End Function

Public Function SetProcessPriority(Optional _
    ByVal ProcessID As Long, Optional ByVal hWnd _
    As Long, Optional ByVal Priority As _
    ProcessPriorities = NORMAL_PRIORITY_CLASS) _
    As Long
    Dim hProc As Long
    Const fdwAccess1 As Long = _
        PROCESS_QUERY_INFORMATION Or _
        PROCESS_SET_INFORMATION
    Const fdwAccess2 As Long = _
        PROCESS_QUERY_INFORMATION

    ' If not passed a PID, then find value from hWnd.
    If ProcessID = 0 Then
        Call GetWindowThreadProcessId(hWnd, _
            ProcessID)

        ' Need to open process with setinfo rights.
        hProc = OpenProcess(fdwAccess1, 0&, ProcessID)
        If hProc Then
            ' Attempt to set new priority.
            Call SetPriorityClass(hProc, Priority)
        Else
            ' Weren't allowed to setinfo, so just open
            ' to enable return of current priority setting.
            hProc = OpenProcess(fdwAccess2, 0&, _
                ProcessID)
        End If

        ' Get current/new setting.
        SetProcessPriority = GetPriorityClass(hProc)
        ' Clean up.
        Call CloseHandle(hProc)
    End Function
```

Listing 1 You might need applications to be responsive to the user immediately, regardless of system load, or you might want them to execute only when the system isn't too busy. Adjusting the overall priority the system assigns your application is as simple as passing a known handle to the routines in this code, along with the desired priority setting. Be sure to read the warnings in the SDK docs about "real-time" processes and their ability to destabilize the system, before you use your desired setting. You should consider even "high" priority usage carefully, as such a setting can consume the CPU fully.

C# • Set Up the ErrorProvider

```
void OnOK(object sender, EventArgs e)
{
```

display an error alert. The icon can blink, and you can configure the blink rate and policy.

Bring up the visual properties designer window for the ErrorProvider control. Here you can provide a custom alert icon

Although ErrorProvider is geared toward validation of data-source bound controls (such as the data grid), you can use it to validate and provide error information on any other control. For example, consider a Windows Forms dialog that asks the user for first and last name (see Figure 1).

The form needs to alert the user if he or she doesn't provide either a first or a last name. Use the ErrorProvider for this by dragging and dropping an ErrorProvider control from the toolbox onto the form. The ErrorProvider isn't a visual control, so it appears underneath the form. The ErrorProvider displays a small icon when you set it to

Microsoft on helping the industry adopt .NET. He's the author of *COM and .NET Component Services* (O'Reilly & Associates). Contact him at www.idesign.net.

**Additional Resources**

SetPriorityClass: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dllproc/prothred_9z1v.asp