ZylIdleTimer 1.50

ZylIdleTimer is a Delphi / C++Builder component which lets you to take actions after a specified system-wide (related to the whole system) or application (related only to the application) idle time or to check the time interval of user inactivity.
System Idle time is the time interval of full user inactivity. In this interval any user action is missing on the computer, like pressing a key of the keyboard or moving the mouse or pressing a mouse button or rotating the scroll.
Application Idle time is the time interval of user inactivity inside of one application. In this interval the user doesn't press a key of the keyboard or a mouse button inside of one window of the application.

The demo version is fully functional in Delphi and C++Builder IDE, but it displays a nag dialog (the licensed version will, of course, not have a nag dialog and will not be limited to the IDE). The package includes demo programs for Delphi and C++Builder and a help file with the description of the component.

Supported Operating Systems:
Windows 2000/XP/Server2003/Vista/7/Server2008/8/Server2012/10/11

Available for:
Delphi 12 (Win32 & Win64), Delphi 11 (Win32 & Win64), Delphi 10.4 (Win32 & Win64), Delphi 10.3 (Win32 & Win64), Delphi 10.2 (Win32 & Win64), Delphi 10.1 (Win32 & Win64), Delphi 10 (Win32 & Win64), Delphi XE8 (Win32 & Win64), Delphi XE7 (Win32 & Win64), Delphi XE6 (Win32 & Win64), Delphi XE5 (Win32 & Win64), Delphi XE4 (Win32 & Win64), Delphi XE3 (Win32 & Win64), Delphi XE2 (Win32 & Win64), Delphi XE, Delphi 2010, Delphi 2009, Delphi 2007, Delphi 2006, Delphi 2005, Delphi 7, Delphi 6, Delphi 5, Delphi 4, C++Builder 12 (Win32 & Win64), C++Builder 11 (Win32 & Win64), C++Builder 10.4 (Win32 & Win64), C++Builder 10.3 (Win32 & Win64), C++Builder 10.2 (Win32 & Win64), C++Builder 10.1 (Win32 & Win64), C++Builder 10 (Win32 & Win64), C++Builder XE8 (Win32 & Win64), C++Builder XE7, C++Builder XE6, C++Builder XE5, C++Builder XE4, C++Builder XE3, C++Builder XE2, C++Builder XE, C++Builder 2010, C++Builder 2009, C++Builder 2007, C++Builder 2006, C++Builder 6, Turbo Delphi, Turbo C++

Remarks:
- The Delphi 2006 version is fully compatible with Turbo Delphi
- The C++Builder 2006 version is fully compatible with Turbo C++

Limitations:
Only one timer with Kind = itApplication can be used for each application

Insatallation:

If you have a previous version of the component installed, you must remove it completely before installing this version. To remove a previous installation, proceed as follows:
-Start the IDE, open the packages page by selecting Component - Install Packages
-Select ZylIdleTimerPack package in the list and click the Remove button
-Open Tools - Environment Options - Library and remove the library path pointing to ZylIdleTimer folder
-Close the IDE
-Browse to the folder where your bpl and dcp files are located (default is $(DELPHI)\Projects\Bpl for Delphi, $(BCB)\Projects\Bpl for C++ Builder). -Delete all of the files related to ZylIdleTimer
-Delete or rename the top folder where ZylIdleTimer is installed

-Unzip the zip file and open the ZylIdleTimerPack.dpk file in Delphi (ZylIdleTimerPack.bpk file in C++Builder), compile and install it
and add to Tools/Environment Options/[Language]/Library (in older Delphi/C++Builder menu) or Tools/Options/Delphi Options/Library/Library Path (in newer Delhi menu) or Tools/Options/C++ Options/Paths and Directories/Library Path & Include Path (in newer C++Builder menu) the path of the installation (where the ZylIdleTimer.dcu file is located). The component will be added to the "Zyl Soft" tab of the component palette. After you have the component on your component palette, you can drag and drop it to any form, where you can set its properties by the Object Inspector and you can write event handlers selecting the Events tab of the Object Inspector and double clicking the preferred event.
If you still have problems in C++Builder, running an application, which contains the component, then open the project and in C++Builder menu, Project/Options/Packages and uncheck "Build with runtime packages".

-It is indicated to use this component with "Stop on Delphi exception" option deactivated. You can do this from Delphi / C++Builder menu, "Tools/Debugger Options/Language Exceptions/Stop on Delphi exceptions", otherwise you will have a break at all the handled exceptions.

64-bit platform:
Delphi/C++Builder 64-bit support is only for runtime, so you have to use it in the folllwing way:
Install the 32-bit version of the component as it described above and add to Tools/Options/Delphi Options/Library/Library Path, selected platform: 64-bit Windows the path of the Win64 subfolder of the component.
Before compiling the host application for 64-bit Windows, right click on Target Platforms, Add Platform and add 64-bit Windows (Make the selected platform active). If you compile the application in this way, it will be a native 64-bit application.

Types:

TIdleTimerKind = (itApplication, itSystem);

Properties:

Enabled - enables or disables the timer
Interval - the idle time interval (system or application level, depending on **Kind**) in milliseconds (1 second = 1000 milliseconds). When this interval is expired then the OnIdle event is fired.
WarningInterval - the warning idle time interval (system or application level, depending on **Kind**) in milliseconds (1 second = 1000 milliseconds). When this interval is expired then the OnWarning event is fired.
IntervalInMinutes - the idle time interval (system or application level, depending on **Kind**) in minutes.
IntervalInSeconds - the idle time interval (system or application level, depending on **Kind**) in seconds.
WarningIntervalInMinutes - the warning idle time interval (system or application level, depending

on **Kind**) in minutes.
WarningIntervalInSeconds - the warning idle time interval (system or application level, depending on **Kind**) in seconds.
Kind - specifies the type of the idle timer. If this property is **itSystem** then **OnIdle** event is fired when the system-wide idle time interval is expired, otherwise this event is fired when the application idle time interval is expired.
Version - returns the current version of the component

Functions:

constructor Create(AOwner: TComponent) - constructor
destructor Destroy - destructor
function GetApplicationIdleTime: Cardinal - if Kind is set to "itApplication", then returns the application idle time in milliseconds, otherwise the result is 0.
function GetSystemIdleTime: Cardinal - returns the system-wide idle time in milliseconds
function GetRelativeSystemIdleTime: Cardinal - returns the minimum between system-wide idle time and the interval from the latest reset in milliseconds.
procedure ResetApplicationIdleTime - sets the application idle time to 0. You can call it in the **OnIdle** event handler to simulate periodical call of this handler till the next user intervention or you can use it to signalize that your application is busy. It works only for **Kind** = itApplication.
procedure ResetSystemIdleTime - sets the system idle time to 0. You can call it in the **OnIdle** event handler to simulate periodical call of this handler till the next user intervention or you can use it to signalize that your system is busy. It works only for **Kind** = itSystem..

Events:

OnIdle: TNotifyEvent - fires when the idle time interval is expired (system or application level depending on **Kind**).
OnResume: TNotifyEvent - fires after the idle time interval is expired (system or application level depending on **Kind**) and the idle time counter is reset due to user intervention.
OnWarning: TNotifyEvent - fires when the warning idle time interval is expired (system or application level depending on **Kind**) .

Buy Now!

Copyright by Zyl Soft 2003 - 2024
http://www.zylsoft.com
info@zylsoft.com