Table of Contents

1 Components Overview

The **CronJob component** is the thread-based alarm implementation for Delphi/BCB which acts like cronjob utility in Unix. The component produces periodical OnAlert events by schedule specified in CRON format (actually it uses extended CRON format with new "seconds" field, to let the event to be triggered even every second).
To specify the moments when the component should trigger the OnAlert event, the component uses the **CRON** format (all who worked in Unix must be aware about this utility, this is daemon of scheduled, periodically executed tasks). The authors of CronJob component has very liked an idea of describing the periodical events of any complexity, tiered to astronomical time, in simple text string. We have used slightly modified CRON format, added seconds field and altered the rules of describing the lists of numbers.

The package includes 2 bonus components:

AccurateTimer — thread-based timer which periodically triggers the OnTimer events by specified Interval, but unlike standard TTimer, which based on Win32API and thus loses timer messages on high application overload, this timer runs in the separate thread and accurately triggers each OnTimer event at right time. It also allows to specify the Interval with higher precision (less than 55msec, limited in Win9x) and does not dependent to the system-wide limitations.)

AwayTimer — monitors user activity and increase its time counter when keyboard and mouse are inactive (user distracted from PC or goes away). Also it triggers special event when system is about to start the screensaver, or after some defined period of user inactivity.

---

2 Installation Instructions

Package without source code

to Delphi 2

1. Unzip files from "Delphi2" directory to your "Delphi 2\Lib" directory.
2. Start Delphi 2 IDE.
3. Select "Component \ Install..." menu item.
4. Press "Add" button and select "CronJob.dcu" file.
5. Rebuild library.

to Delphi 3

1. Unzip files from "Delphi3" directory and copy them to "Delphi 3\Lib".
2. Start Delphi 3 IDE.
3. Open "CronJobD3.dpk" file.
4. Install package to the components palette ("Install" button).

to Delphi 4

1. Unzip files from "Delphi4" directory and copy them to "Delphi 4\Lib".
2. Start Delphi 4 IDE.
3. Open "CronJobD4.dpk" file.
4. Install package to the components palette ("Install" button).

to Delphi 5

1. Unzip files from "Delphi5" directory and copy them to "Delphi 5\Lib".
2. Start Delphi 5 IDE.
3. Open "CronJobD5.dpk" file.
4. Install package to the components palette ("Install" button).

to Delphi 6

1. Unzip files from "Delphi6" directory and copy them to "Delphi 6\Lib".
2. Start Delphi 6 IDE.
3. Open "CronJobD6.dpk" file.
4. Install package to the components palette ("Install" button).

to Delphi 7

1. Unzip files from "Delphi7" directory and copy them to "Delphi 7\Lib".
2. Start Delphi 7 IDE.
3. Open "CronJobD7.dpk" file.
4. Install package to the components palette ("Install" button).

to C++ Builder 1

1. Unzip files from "BCB1" directory to your "CBuilder\Lib" directory.
2. Start C++ Builder IDE.
3. Select "Component \ Install..." menu item.
4. Press "Add" button and select "CronJob.dcu" file.
5. Rebuild library.

to C++ Builder 3

1. Unzip files from "BCB3" directory and copy them to "CBuilder3\Lib".
2. Start C++ Builder 3 IDE.
3. Open "CronJobCB3.bpk" file.
6. Select "Project \ Make CronJobCB3" menu item.
7. Select "Component \ InstallPackages" menu item.
8. Press "Add" button and select "CronJobCB3.bpl" file.

to C++ Builder 4
1. Unzip files from "BCB4" directory and copy them to "CBuilder4\Lib".
2. Start C++ Builder 4 IDE.
3. Open "CronJobCB4.bpk" file.
4. Install package to the components palette ("Install" button).

to C++ Builder 5
1. Unzip files from "BCB5" directory and copy them to "CBuilder5\Lib".
2. Start C++ Builder 5 IDE.
3. Open "CronJobCB5.bpk" file.
4. Install package to the components palette ("Install" button).

to C++ Builder 6
1. Unzip files from "BCB6" directory and copy them to "CBuilder6\Lib".
2. Start C++ Builder 6 IDE.
3. Open "CronJobCB6.bpk" file.
4. Install package to the components palette ("Install" button).

Source code
1. Uninstall / delete all previous (trial) instances of CronJob.
2. Unzip files from "Sources" directory and copy them to "..\Lib" directory.
3. Run Delphi or ++ Builder IDE.
4. Select "Component \ Install..." menu item.
5. Press "Add" button and select "CronJob.pas" file.
6. Rebuild library.

3 Registration Information

CronJob component is SHAREWARE. This means that you can try it out for free, but if you like it and want to use it you have to register it with the author. Before continue read and accept license agreement please.

The only difference between the unregistered and registered versions is that the registered one has not message box with remind to register and works without Delphi (C++ Builder) running. You can also purchase the source code, if you would like to have it, and be able to compile or modify the CronJob on any 32bit version of Delphi or C++ Builder.

If you would like to use the CronJob and receive full, unrestricted version, priority support or even source code — you have to purchase proper license.

All prices in US dollars. Registering entitles you to unlimited support via E-Mail, minor version updates indefinitely and major version updates for 6 month from date of purchase. You can use registered components in any number of projects, there is no deployment and royaltee fees.

Registration types:

Full, unrestricted version without source code:
Single user license:
- **https://secure.element5.com/register.html?productid=210236 - $14,95**

Site license:
- https://secure.element5.com/register.html?productid=210238 - $59,95

Full version including 100% Source Code:

Single user license:
- **https://secure.element5.com/register.html?productid=210239 - $29,95**

Site license:
- https://secure.element5.com/register.html?productid=210240 - $89,95

Comments

1. ***Site license*** covers a single organisation in one location (building complex). If you buy a site license, you may use the software in unlimited number of your company's computers withing this area. Site license is very cost-effective if you have many computers (many software developers).

See license agreement for more details.

CronJob component (http://www.appcontrols.com)

Documentation created with **Help&Manual**, best authoring tool.

4 License Agreement

Copyright

The CronJob component (software) is Copyright © 2003-2004, by Utilmind Solutions® (Utilmind). All rights reserved.
The authors - Utilmind Solutions® and Aleksey Kuznetsov (founder of Utilmind), exclusively own all copyrights to the Advanced Application Controls (AppControls) and all other products distributed by Utilmind Solutions®.

Liability disclaimer

THIS SOFTWARE IS DISTRIBUTED "AS IS" AND WITHOUT WARRANTIES AS TO PERFORMANCE OF MERCHANTABILITY OR ANY OTHER WARRANTIES WHETHER EXPRESSED OR IMPLIED. YOU USE IT AT YOUR OWN RISK. THE AUTHOR WILL NOT BE LIABLE FOR DATA LOSS, DAMAGES, LOSS OF PROFITS OR ANY OTHER KIND OF LOSS WHILE USING OR MISUSING THIS SOFTWARE.

Restrictions

You may not attempt to reverse compile, modify, translate or disassemble the software in whole or in part. You may not remove or modify any copyright notice or the method by which it may be invoked.

Operating license

Unregistered version

You may distribute the unregistered version of software freely, provided that all files are included and remain unmodified and that no extra files have been added to the package. You may not ask any money for the distribution. You may use the unregistered version of software free of charge for testing purposes, but if you want to use it for other purposes than testing - you have to register it with the author.

Registered version (single user license)

Once you have registered, you will receive a personal registered copy via email and login information to access your personal area at AppControls.com. This copy may not be copied or lend. You have the non-exclusive right to use registered version of the software only by a single person, on a single computer at a time. You may physically transfer the software from one computer to another, provided that the software is used only by a single person, on a single computer at a time.

In group projects where multiple persons will use the software, you must purchase an individual license for each member of the group or purchase site license. Use over a "local area network" (within the same locale) is permitted provided that the software is used only by a single person, on a single computer at a time. Use over a "wide area network" (outside the same locale) is strictly prohibited under any and all circumstances.

Registered version (site/team license)
Once you have registered, you will receive a personal registered copy via email and login information to access your personal area at AppControls.com. This copy may not be copied or lend. You have the non-exclusive right to use and transfer registered version of software on any number of computers by your company or your team only in one location (building complex). If you purchase a site license, you may use the program in an unlimited number of your company's computers within this area.

Registered version (Educational site license)
Once you have registered, you will receive a personal registered copy via email and login information to access your personal area at AppControls.com. This copy may not be copied or lend. You have the non-exclusive right to use and transfer registered version of software on any number of computers by your educational organisation (school/college/university etc) in one location (building complex). If you buy a educational site license, you may use the program in an unlimited number of your edicational organisation's computers within this area.

Registered version (World-wide license)
Once you have registered, you will receive a personal registered copy via email and login information to access your personal area at AppControls.com. This copy may not be copied or lend. You have the non-exclusive right to use and transfer registered version of software on any number of computers by your company or your team world-wide. If your company has many branches even with thouthands of computers, world wide license covers them all.

Notes (clarification)
"Single-user license" means "single-developer license". "Site license" means that it can be used by any number of software developers within your company.
You can use purchased components in ANY number of your projects and deploy the "end-user" software to ANY number of your users/customers without any additional royalty fees. However you are not permitted to distribute the component itself (the source code or .dcu files of components).

Back-up and transfer
You may make one copy of the software solely for "back-up" purposes, as prescribed by international copyright laws. You must reproduce and include the copyright notice on the back-up copy.

Terms
This license is effective until terminated. You may terminate it by destroying the program, the documentation and copies thereof. This license will also terminate if you fail to comply with any terms or conditions of this agreement. You agree upon such termination to destroy all copies of the program and of the documentation, or return them to author.

Other rights and restrictions
All other rights and restrictions not specifically granted in this license are reserved by authors.

CronJob component (http://www.appcontrols.com)

5 CronJob component

5.1 TumCronJob

Overview

The CronJob component is the thread-based alarm timer implementation for Delphi and C++ Builder which acts like *cronjob* utility in Unix. The component produces periodical OnAlert events by schedule specified in CRON format (actually it uses extended CRON format with new "seconds" field, to let the event to be triggered even every second).

CRON format

To specify the moments when the component should trigger the OnAlert event, the component uses the **CRON** format (all who worked in Unix must be aware about this utility, this is daemon of scheduled, periodically executed tasks). The authors of CronJob component has very liked an idea of describing the periodical events of any complexity, tiered to astronomical time, in simple text string. We have used slightly modified CRON format, added seconds field and altered the rules of describing the lists of numbers.

The CronMask property presents itself of several "list of numbers", separated by space. Every list specifies enumeration of moments of time or date, in units, dependant of position of the list in the mask string.

The CronMask schedule should be specified in following maneer (list of numbers/enumerations separated by spaces):

```
Seconds Minutes Hours Days Months DaysOfWeek
```

Here is some examples of describing of schedule in the CRON format:

- Each minute at 0 seconds and 30 seconds:
```
0,30
0+30
+30
```
- Each second at 0, 8 and 16 hours:
```
0-59 0-59 0-16+8
* * 0+8
* * +8
```
- The beginning of each hour, except noon and midnight:
```
0 0 1-11,13-23
```
- Every 3 seconds of 1st day of each month:
```
0-59+3 * * 1
+3 * * 1
```
- 30 minutes, 0 seconds of each hour at Sunday:
```
0 30 * * * 0
```

Description of CRON lists

The string list with numbers without spaces is the group of numbers separated by comma "," (i.e: `0,30`). The group can contain one number or the range of numbers. The ranges should be specified by two numbers, separated by hyphen "-" (the beginning and ending of the range, i.e: `0-9`). Optionally, after the range you can also specify the value of *step,* separated by plus "+" (i.e: `0-59+5`). The default step value (if none specified) is 1. If the step specified, the ending value may be not specified, by default it will be the maximal value in context of purpose of the list. The beginning of the range is 0 by default (or 1 for days and monthes).

The asterisk sign "*" instead of group means all diapasone of possible values in the context.

The order of values or ranges in one list is not important. For example, the list like `0-5,12,8,20-`

`30+2` is interpreted as following enumeration: `0,1,2,3,4,5,8,12,20,22,24,26,28,30`.

How to use
Drop the CronJob component on your form, specify the CronMask property and write the OnAlert event handler which will process scheduled events.

See also
umAccurateTimer and umAwayTimer components.

5.2 Properties

5.2.1 Active

Applies to
umCronJob, umAwayTimer and umAccurateTimer components.

Declaration
```
property Active: Boolean; // True by default
```

Description
The Active property controls whether the umCronJob is active and able to trigger OnAlert event at the time scheduled in CronMask property.

Set Active property to True to enable the cronjob and, othwise set it to False to disable it.

See also
CronMask property and OnAlert event.

5.2.2 AlertsCount

Applies to
umCronJob and umAccurateTimer components.

Declaration
```
property AlertsCount: Cardinal; // read-only!
```

Description
The AlertsCount is the read-only property which determines the number triggered OnAlert events in current application session.

See also
AlertsCount property;
OnAlert event.

5.2.3 CronMask

Applies to
umCronJob component.

Declaration
```
property CronMask: String;
```

Description
The CronMask string specifies the string in CRON format, which describes scheduled time periods to trigger OnAlert events.

CRON format

To specify the moments when the component should trigger the OnAlert event, the component uses the **CRON** format (all who worked in Unix must be aware about this utility, this is daemon of scheduled, periodically executed tasks). The authors of CronJob component has very liked an idea of describing the periodical events of any complexity, tiered to astronomical time, in simple text string. We have used slightly modified CRON format, added seconds field and altered the rules of describing the lists of numbers.

The CronMask property presents itself of several "list of numbers", separated by space. Every list specifies enumeration of moments of time or date, in units, dependant of position of the list in the mask string.

The CronMask schedule should be specified in following maneer (list of numbers/enumerations separated by spaces):

```
Seconds Minutes Hours Days Months DaysOfWeek
```

Here is some examples of describing of schedule in the CRON format:

- Each minute at 0 seconds and 30 seconds:
```
0,30
0+30
+30
```
- Each second at 0, 8 and 16 hours:
```
0-59 0-59 0-16+8
* * 0+8
* * +8
```
- The beginning of each hour, except noon and midnight:
```
0 0 1-11,13-23
```
- Every 3 seconds of 1st day of each month:
```
0-59+3 * * 1
+3 * * 1
```
- 30 minutes, 0 seconds of each hour at Sunday:
```
0 30 * * * 0
```

Description of CRON lists

The string list with numbers without spaces is the group of numbers separated by comma "," (i.e: `0,30`). The group can contain one number or the range of numbers. The ranges should be specified by two numbers, separated by hyphen "-" (the beginning and ending of the range, i.e: `0-9`). Optionally, after the range you can also specify the value of *step,* separated by plus "+" (i.e: `0-59+5`). The default step value (if none specified) is 1. If the step specified, the ending value may be not specified, by default it will be the maximal value in context of purpose of the list. The beginning of the range is 0 by default (or 1 for days and monthes).

The asterisk sign "*" instead of group means all diapasone of possible values in the context.

The order of values or ranges in one list is not important. For example, the list like `0-5,12,8,20-30+2` is interpreted as following enumeration: `0,1,2,3,4,5,8,12,20,22,24,26,28,30`.

See also

Active property and OnAlert event.

5.2.4 LastTime

Applies to

umCronJob component.

Declaration

```
property LastTime: TDateTime; // read-only!
```

Description

The LastTime is read-only property which determines the date/time of last triggered OnAlert event.

See also
AlertsCount property.

5.3 Events

5.3.1 OnAlert

Applies to
umCronJob component.

Declaration
```
property OnAlert: TNotifyEvent;
```

Description
The OnAlert event occurs at the time described in CronMask property. Write this event handler to take some specific actions by specified schedule.

See also
Active and CronMask properties.

6 AccurateTimer component

6.1 TumAccurateTimer

Overview
The AccurateTimer component is the thread-based timer which periodically triggers the OnTimer events by specified Interval. Unlike standard TTimer, which based on Win32API and thus loses timer messages on high application overload, this timer runs in the separate thread and accurately triggers each OnTimer event at right time. It also allows to specify the Interval with higher precision (less than 55msec, limited in Win9x) and does not dependent to the system-wide limitations.

How to use
Drop the AccurateTimer on your form, specify the Interval (amount of time, in milliseconds, that passes before the component initiates another OnTimer event), and write the OnTimer event handler to execute an action at regular intervals.

See also
umCronJob and umAwayTimer components.

6.2 Properties

6.2.1 Active

Applies to
umAccurateTimer, umAwayTimer and umAwayTimer components.

Declaration
```
property Active: Boolean; // True by default
```

Description
The Active property controls whether the umAccurateTimer is active and periodically triggers OnTimer events, with timer interval specified in the Interval property.

Set Active property to True to enable the timer, othwise set it to False to disable it.

See also
Interval property and OnTimer event.

6.2.2 Interval

Applies to
umAccurateTimer component.

Declaration
```
property Interval: Integer;
```

Description
The Interval property determines the amount of time, in milliseconds, that passes before the timer component initiates another OnTimer event.

Interval determines how frequently the OnTimer event occurs. Each time the specified interval passes, the OnTimer event occurs.

Use Interval to specify any cardinal value as the interval between OnTimer events. The default value is 1000 (one second).

 Use Reset method to reset the timer and start counting the amounts of time from current moment.

See also
Active property;
Reset method;
OnTimer event.

6.3 Methods

6.3.1 Reset

Applies to
umAccurateTimer component.

Declaration
```
procedure Reset;
```

Description
The Reset method resets the timer and start counting the intervals from the current moment. After calling the Reset method, the next OnTimer event will occur after passing the amount of time specified in the Interval property.

See also
Active and Interval properties;
OnTimer event.

6.4 Events

6.4.1 OnTimer

Applies to
umAccurateTimer component.

Declaration
```
property OnTimer: TNotifyEvent;
```

Description
The OnTimer event occurs when a specified amount of time, determined by the Interval property, has passed. Write OnTimer event handler to execute an action at regular intervals.

The Interval property of a timer determines how frequently the OnTimer event occurs. Each time the specified interval passes, the OnTimer event occurs.

Use Reset method to reset the timer and start counting the amounts of time from current moment.

See also
Active and Interval properties;
Reset method.

7 AwayTimer component

7.1 TumAwayTimer

Overview
The umAwayTimer component monitors user activity and increase its time counter when keyboard and mouse are inactive (user distracted from PC or goes away). Also it triggers special event when system is about to start the screensaver, or after some defined period of user inactivity.

Advantages ?
You can execute some specific actions when user away from PC. For example, automatically save file, re-index the database etc. If you are writing Internet communication tool, you can switch the status of presence to "inactive" (Sleep/Away mode, like it does most Inernet Conferencing and Instant Messenger tools like ICQ, AOL, MSN or WinJabber), to inform user on the other side that user is temporary away.

How does it works and how to use ?
When the Active property is True, the umAwayTimer checks the mouse movements and keyboard presses with time interval specified in MonitorInterval property (in milliseconds).

When component does not detects any activity (user doesn't use mouse and keyboard for some time), it increases special time counter which can be used to determinate how much time user was inactive (temporary away from PC, went to smoke, have break for coffee etc). You can determinate the time of inactivity reading AwayTime property. This is read-only property, however if you want to reset the AwayTime property programmatically — use ResetAway method.

If you want to perform some specific actions after some time of inactivity — specify required time period (in units of milliseconds) to SleepInterval property and write OnSleep event handler. To be notified when user became active again (for example to return "Away" status to previous) — write OnAwake even handler.

Also, the umAwayTimer detects when the screen saver starts and become deactivated. To be notified about screen saver presence — write OnScreenSaverStart and OnScreenSaverEnd events.

See also
umAccurateTimer and umCronJob components.

7.2 Properties

7.2.1 Active

Applies to
umAwayTimer, umAccurateTimer and umCronJob components.

Declaration
```
property Active: Boolean; // True by default
```

Description
The Active property controls whether the umAwayTimer is active to monitor the system for user activity.

Set Active property to True to enable the timer and check system for mouse movements and keyboard state, othwise set it to False to disable the monitoring timer.

 The detection are performed with intervals specified in MonitorInterval property.

See also
MonitorInterval property.

7.2.2 AwayTime

Applies to
umAwayTimer component.

Declaration
```
property AwayTime: Cardinal; // read-only. time in milliseconds
```

Description
The AwayTime is the read-only property used to determinate the time interval (in milliseconds) passed after last user activity (i.e: number of milliseconds passed after last mouse movement or last key press).

See also
MonitorInterval property and ResetAway method.

7.2.3 MonitorInterval

Applies to
umAwayTimer component.

Declaration
```
property MonitorInterval: Integer; // 100 milliseconds by default
```

Description
The MonitorInterval property specifies the time intervals, in units of milliseconds, between monitoring the system for mouse movements and keyboard state, if Active property is True.

By default, the MonitorInterval set to 100, that means that component is checking the user activity every 100 milliseconds (10 times per second).

See also
Active property.

7.2.4 SleepInterval

Applies to
umAwayTimer component.

Declaration
```
property SleepInterval: Cardinal; // 5 minutes by default
```

Description
The SleepInterval property specifies the time interval (in milliseconds), which should pass since last user activity till execution of the OnSleep event.

For example, the OnSleep event will occur in 5 minutes of user inactivity (if user did not moved mouse and did not pressed keyboard buttons), if SleepInterval is 300000 (5 minutes = 300 seconds = 300000 milliseconds).

See also
OnSleep event.

7.3 Methods

7.3.1 ResetAway

Applies to
umAwayTimer component.

Declaration
```
procedure ResetAway;
```

Description
The ResetAway method reduces the AwayTime variable to zero.

See also
AwayTime property.

7.4 Events

7.4.1 OnActiveTimer

Applies to
umAwayTimer component.

Declaration
```
property OnActiveTimer: TNotifyEvent;
```

Description
The OnActiveTimer event is like OnTimer event of usual TTimer component, but occurs only when user actively using his PC (until umAwayTimer component not detected that user "sleeping").

This event allows to perform some specific actions, with interval specified in MonitorInterval property, but ONLY when user IS ACTIVE (for example you can show him some popup ads or something :-)). This event never occurs when computer is inactive.

See also
Active and MonitorInterval property.

7.4.2 OnScreenSaverStart

Applies to
umAwayTimer component.

Declaration
```
property OnScreenSaverStart: TNotifyEvent;
```

Description
The OnScreenSaverStart event occurs when the component detects that screen saver program became active.

Example *(this will play sound on starting the screen saver)*
```
procedure TForm1.umAwayTimer1ScreenSaverStart(Sender: TObject);
begin
  acWavPlayer1.Play;
end;
```

See also
OnScreenSaverEnd event.

7.4.3 OnScreenSaverEnd

Applies to
umAwayTimer component.

Declaration
```
property OnScreenSaverEnd: TNotifyEvent;
```

Description
The OnScreenSaverEnd event occurs when the component detects that screen saver program was deactivated by user.

See also
OnScreenSaverStart event.

7.4.4 OnSleep

Applies to
umAwayTimer component.

Declaration
```
property OnSleep: TNotifyEvent;
```

Description
The OnSleep event occurs after some period of user inactivity, specified in SleepInterval property.

For example, the OnSleep event will occur in 5 minutes of user inactivity (if user did not moved mouse and did not pressed keyboard buttons), if SleepInterval property is 300000 (5 minutes = 300

seconds = 300000 milliseconds).


You can use OnSleep event to execute some specific actions when user away from PC. For example, automatically save file, re-index the database etc. If you are writing Internet communication tool, you can switch the status of presence to "inactive" (Sleep/Away mode, like it does most Inernet Conferencing and Instant Messenger tools like ICQ, AOL, MSN or WinJabber), to inform user on the other side that user is temporary away. If you wish to return the presence status to previous state when user become active — use OnAwake event handler.

See also
SleepInterval property and OnAwake event.

7.4.5 OnAwake

Applies to
umAwayTimer component.

Declaration
```
property OnAwake: TNotifyEvent;
```

Description
The OnAwake event occurs when user awake (become active) after some period of inactivity, specified in SleepInterval property.


Use OnAwake event to return the application status to previous state after "sleeping" (Away mode). To be notified when user inactive after some time period — write OnSleep event handler.

See also
SleepInterval property and OnSleep event.