

Q: I want my application to periodically and automatically perform an operation. For instance, I want to save a backup file every 5 minutes, or I want to create a blinking cursor. How do I go about doing that? Should I use timer events?

A: What you actually want is a timed entry. The Display PostScript routines **DPSAddTimedEntry()** and **DPSRemoveTimedEntry()** are used to start and stop timed entries. These functions are described in the Next Developer Documentation in the DPS section. You install a timed entry to run at a specific interval and give a handler function which will be called at each interval. What actually happens is that for each cycle of the event loop, the application checks to see if there are any timed entries which are due. If so, then the handler function is called, thus "cutting" in line in front of any other events. Timed entries are ^acoalesced^o in a sense because the next occurrence of a timed entry is set when the current entry is processed. There is never more than one occurrence of a timed entry waiting to run and since timed entries aren't inserted into the event queue, you don't have to worry about timed entries overflowing the queue.

Timer events are not the correct thing to use in this instance, because they are intended to be used in modal loops. Each application has an event queue, where pending events are waiting to be processed. Each application has a main loop which simply polls for events, and then reacts accordingly. Modal loops allow you to create a loop which is secondary to the application loop, and supersedes it for a short time. You use modal loops when you have received one event and are then waiting for another which then terminates the modal loop. The modal loop must ensure that it will continue to get a continuous stream of events, and it does this with timer events. You use timer events

in a modal loop in case the user is doing something that is not generating events (such as holding down the mouse button, and not moving it).

There are several programming examples that use timed entries. If you want a simple example of a timed entry, see the **BusyBox** example under **/NextDeveloper/Examples/AppKit**. The **ClockView** class there uses a timed entry to animate the movement of the hands. (Tip: It can be very helpful to drag **/NextDeveloper/Examples** into your Digital Librarian and index it. Another useful Librarian target is **/usr/include**.)

Valid for 1.0, 2.0, 3.0

QA650

Q: I've installed a timed entry to run at a specific interval in my application. I receive a timed entry and my application goes off to process it. For some reason the processing takes longer than the interval between the timed entries. The result is that a second entry happens before the application finishes processing the previous entry. Will the second entry be queued or does it interrupt?

For Releases 1 and 2:

A: Neither. The timed entry interval specifies the time that passes between the time the timed entry function returns and the time it is called again. Thus, if you have an interval of 10 seconds, and your timed entry function takes 5 seconds to execute, your function is called every 15 seconds.

Sometimes this might be a bad thing; it is then the responsibility of your timed entry function to adjust the interval. For instance, in the ClockView class (/NextDeveloper/Examples/Clock under 1.0 and /NextDeveloper/Examples/BusyBox under 2.0), the function that gets called at the top of the minute stops the timed entry and starts it again with an new interval equivalent to the number of seconds left to the next top of the minute. This prevents the clock from ^aslowing down^o and missing minutes when the system is slow. (It however does not bother with this in the seconds mode; missing a second or two here and there is okay.)

The Animator class shows how to create an ^aadjusting^o timed entry, which is a bit more complex. The Animator class is used by /NextDeveloper/Examples/BreakApp under Release 1 and /NextDeveloper/Examples/ToolInspector under Release 2.

For Release 3:

A: In Release 3, the system tries to call the function with the requested periodicity, regardless of how long the function takes to execute. However, if the function takes longer than the period to execute, the timed entries do not try to "catch up" to make up for the missed call(s).

In Releases 1 and 2, timed entries were accurate to only 15 ms. In Release 3, they are accurate to ~1 ms.

QA349

Valid for 1.0, 2.0,r 3.0