

AcroTeX.Net

**The cntdwn Package**  
**Handling Notification Events for the Long**  
**Countdown Timer**

D. P. Story

## 1. Introduction

The example in this section was one I developed to test the notification events of a long countdown. The long countdown is, well, long; the example contained herein counts down 20 seconds, and during that time the various notification events occur. A custom event handler is used to test the long countdown mechanism, and to illustrate not to write such a handler.

## 2. Custom Long Countdown Event Handler

As with an earlier [AeB Blog](#) article in this series, “The cntdwn Package: Handling Notification Events for the Short Countdown Timer,” [scntdwn\\_events.pdf](#), with begin with an example, then discuss how this example was created.

Start the countdown by pressing the square button below. The countdown only lasts 35 seconds.

The AcroTeX Party:

Let's begin the discussion of the code by looking the `\setLongCntDwn` declaration in the preamble:

```

1  \setLongCntDwn{AcroTeXParty}{%
2      autorun=false,
3      date=2010,time=12,
4      notify1=15*\seconds,notify2=10*\seconds,
5      notify3=5*\seconds,notify5=5*\seconds,
6      notify6=10*\seconds,notify7=15*\seconds,
7      eventhandler=myEventHandler,
8      endtimecolor=color.blue
9  }
```

In line (2) we set `autorun=false`, we are going to have a custom button to start the count. The `date` and `time` entries in line (3) are not important because the button is going to dynamically seed these values. `notify1–notify7` are set to 5 second time intervals around time 0. The new part is in line (7) where we declare our custom event handler, `eventhandler=myEventHandler`, to be discussed in what follows. Finally, customize the `endtimecolor` key to blue (the default is red); when count reaches 0, the display changes to this color.

The custom event handler `myEventHandler` is defined in the preamble as well. Below is its definition along with comments.

```

1  \begin{insDLJS}[myEventHandler]{dps}{My Event Handlers for the party!}
2  var _endMsg="End of the party and the count";
3  function myEventHandler (doc,cTimer,nEvent) {
4      var f=doc.getField("AcroTeXPartyMsgs");
5      switch(nEvent) {
6          case 1: f.value+="1. Hurry! Only 15 seconds left!\r";
7                  break;
8          case 2: f.value+="2. 10 seconds left to get to the event of the day!\r";
9                  break;
10         case 3: f.value+="3. Get ready to party at AcroTeX headquarters!\r";
11                break;
12         case 4: f.value+="4. Let's Party! AcroTeX rocks the world!\r";
13                break;
14         case 5: f.value+="5. The party has started, you are missing all the fun, "
15                 + "but it is still not too late to get here.\r";
16                break;
17         case 6: f.value+="6. The party started 10 seconds ago, you can still "
18                 + "get inside if you bring donuts and milk!\r";
19                break;
20         case 7: f.value+="7. You just missed the greatest party, but no worry, "
21                 + "another starts when you press the start button!"
22                 + "\ncntdwnPause(_oAcroTeXParty);
23                 + global.oField=doc.getField("_oAcroTeXParty.lcntdwn.timeToFromEvent");
24                 + var _to=app.setTimeout("var v=function(f){f.value=_endMsg};"
25                 + "v(global.oField);",1000);
26         }
27     }
28 \end{insDLJS}

```

The event handler takes three arguments `doc`, `cTimer`, and `nEvent`; these are the document object, the name of the timer, and the event number, respectively. The events are numbered 1–7, event 4 is the event of reaching 0 on the timer.

In line (4), we get the field object of the text field that hold the various messages. In lines (6)–(21) we concatenate a new string onto the value of the field.

Lines (2)–(25) are part of case 7, the last notification event, are some extra effects added. After writing the last phrase to the text field, we pause the count `lcntdwnPause(_oAcroTeXParty)` in line (22). We then get the field object of the display text field, and assign it as a global variable in line (23). Finally, we execute `app.setTimeout` to write a message to the display after 1 second. The use of `app.setTimeout` is need to wait for the counter to stop counting.

Lines (22)–(25) are icing on the cake, they are optional.

The only other thing to do is to discuss the JavaScript of the button. As I said earlier, this example was a test example I created to test notification events, normally, such a button is not needed.

The button looks like this

```

\pushButton[\TU{Start countdown}\A{\JS{\strLngCntJS}}
]{specialStart}{11bp}{11bp}\kern1bp

```

The command `\strLngCntJS` was defined in the preamble and expands to the JavaScript code. Let's look at its definition.

```
1 \begin{defineJS}{\strLngCntJS}
2   lcntdwnPause(_oAcroTeXParty);
3   this.getField("AcroTeXPartyMsgs").value="";
4   var f=this.getField("_oAcroTeXParty.lcntdwn.timeToFromEvent");
5   f.value="";
6   f.textColor=color.black;
7   var d=new Date();
8   d.setSeconds(d.getSeconds()+20);
9   d=new Date(d.getTime());
10  _oAcroTeXParty.pdfdate=util.printd("D:yyyymmddHHMMss", d );
11  lResetNotify(_oAcroTeXParty);
12  lStartTimer(_oAcroTeXParty);
13 \end{defineJS}
```

The code is defined through the `defineJS` environment of the `insdljs` package; alternatively, it could have been defined as a JavaScript function and included in the `insDLJS` environment.

**Comments:** In line (2) we pause the count, and in lines (2)–(5) we clear the display field and the message field. In line (6) we set the text color of the display field back to black.

Lines (7)–(10) set the date of the countdown to 20 seconds in the future (from the time the code is executed). In line (7), we get the current date, and advance the seconds by 20 in line (8). We then create a new date, also denoted by `d` in line (9). Line (10) is the trick part. The `cntdwn` package holds the date/time information in the form of a PDF date string.<sup>1</sup> In line (10), we set the `pdfdate` property of the `_oAcroTeXParty` object (the object that holds all the timer information) to the new date value, written in the expected format, using the `util.printd` method of Acrobat JavaScript. The last two lines are simple now, line (11) we reset all the Boolean notification switches back to their default values of `false` (defined in the preamble of this document), and then call `lStartTimer`, a function defined in the `cntdwn` package, to start the count back up again.

The code, as just explained, gives hints to the techniques needed to dynamically control the long counter.

That's it for now, hope you didn't fall asleep during the long countdown. ☺

---

<sup>1</sup>The PDF Date Format is specified in section 3.8.3 in the PDF Reference.