DerbyNet Race Timer Operation

Table of Contents

The In-Browser Timer Interface....2

Caveat....2

The derby-timer.jar Option....2

Supported Timers....4

Support for Other Timers and Gate Releases....4

Automatic Gate Releases....5

Configuration for MicroWizard Light Tree & Automatic Gate Release....5

Troubleshooting and Timer Log Files....5

Can’t Run derby-timer.jar on macOS....5

Unable to connect to hosted instance: "ValidatorException"....5

Scanning Runs Without Detecting Timer....5

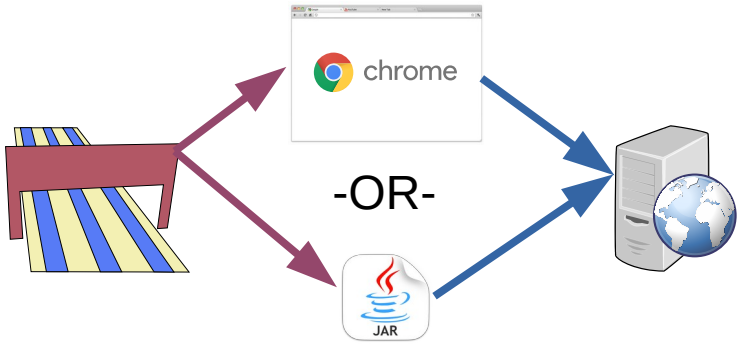
Getting Log Files....6

Server-Side Timer Log Files....6

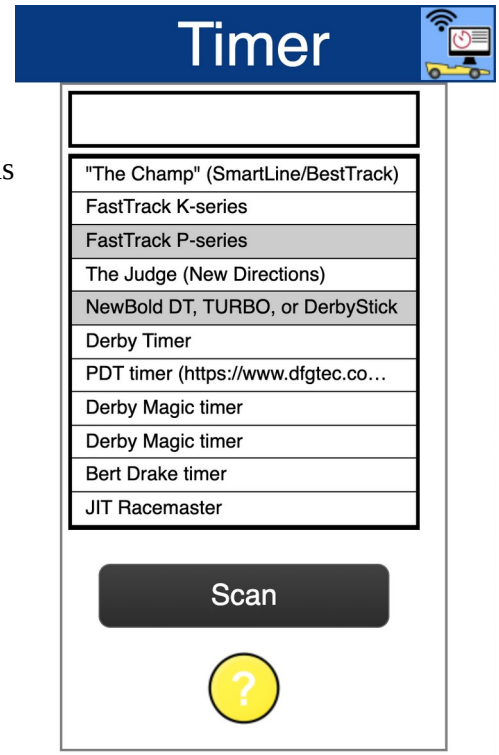
Local Log Files With derby-timer.jar....6

Communication between your track timer and the DerbyNet server allows a much more seamless racing experience. Starting with DerbyNet v7.0, there are several distinct mechanisms by which this communication can be established.

- DerbyNet includes an interface page that can communicate directly with the track timer from a browser, provided certain requirements are met.
- Otherwise, you can run a small Java program, derby-timer.jar, on whatever machine your timer is connected to. This option requires that Java be installed on the machine. Most but not all machines will already have Java installed.



The In-Browser Timer Interface



The in-browser timer interface allows the browser to communicate directly with the race timer, provided these two conditions are met:

1. You're using a browser that supports the new Web Serial API. As of this writing, current versions of Chrome (https://www.google.com/chrome/) include support for this API.)

2. Your browser is either running on the same machine as the DerbyNet server, or is connected to the server using a secure (https) connection[1].

From the Race Dashboard, in the timer status area, the "Timer" button will launch the in-browser timer interface in a separate window.

To use the in-browser timer interface, click the "Scan" button. You will be presented with a list of available serial ports on the machine. You must explicitly grant access to each serial port to which your track timer might be connected.

If you're using a track timer that's not automatically detectable (shown with a grey background), you'll need to click the entry that describes your timer.

Caveat

Some users have reported that the in-browser timer interface may slow or stop functioning when the interface windows doesn't have focus. This behavior varies by browser, and appears to be an attempt to reduce battery usage for mobile devices. It appears not to be something that's controllable by the application software (i.e., DerbyNet).

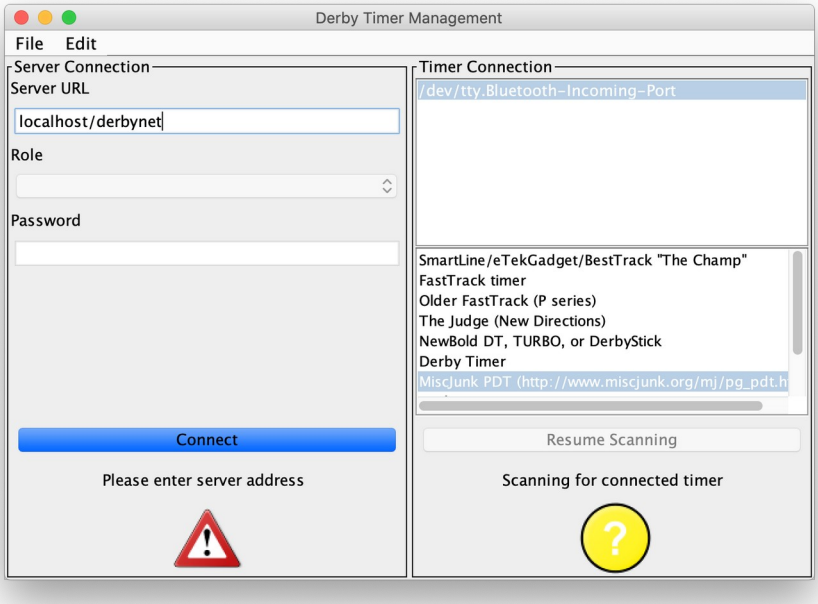
If you notice this issue, you'll need to click on the timer window periodically to restore focus. This is obviously not ideal, so you might wish to consider switching to derby-timer.jar. Either way, do please be in touch if you're affected by this behavior.

The derby-timer.jar Option

On most platforms, derby-timer.jar can be launched by double-clicking, or from the command line. (If opening derby-timer.jar results in a "How do you want to open this file?" message, or something similar, you may need to install a Java runtime; visit java.com for details on installation.)

1See https://developer.mozilla.org/en-US/docs/Web/Security/Secure_Contexts for more information about "Secure Contexts."

When launched, derby-timer.jar puts up a simple user interface, seen here:



The left side of the window controls the connection to the DerbyNet web server. If you run derby-timer.jar on the machine that is hosting your DerbyNet web server, enter **localhost** as the server URL, and click “Connect.”

The right side of the window controls the serial port connection to the timer device. derby-timer.jar will attempt to scan all available serial ports and identify the attached timer device. If this scan doesn’t successfully identify your timer, click on the correct serial port and timer device to force the choice. See also the “-d” and “-n” command-line options.

For automation purposes, or for platforms that don’t support launching a jar file by double-clicking, derby-timer.jar can be launched from the command line:

```
java -jar derby-timer.jar –help
```

(Note that, when run from the command line, it’s possible to suppress the user interface and run derby-timer.jar completely headless.)

For a compatible but not discoverable timer, you may prefer to identify the timer and the port from the command line rather than the user interface, e.g.,

```
java -jar derby-timer.jar -n /dev/usb.serial -d OlderFastTrackDevice
```

Use the –help option to see a list of known devices; choose the name that corresponds to your timer.

Supported Timers

The following electronic timers are automatically detectable:

- MicroWizard "Fast Track" K-Series or Q-Series Timer: http://www.microwizard.com
- SmartLine / eTekGadget / BestTrack "The Champ" Timer: http://www.besttrack.com/champ_timer.htm
- "The Judge:" http://www.newdirections.ws
- The Arduino-based PDT timer formerly known as "MiscJunk" (http://dfgtec.com/pdt)
- Derby Timer: https://derbytimer.com
- Derby Magic timer: http://www.derbymagic.com[2]
- Bert Drake timer (http://drakedev.com/pinewood/)
- JIT Racemaster
- SuperTimer II: http://www.supertimer.com (untested[3])

Additionally, the following timers are compatible, but can't be automatically discovered. Instead, they must be manually specified on the command line, or by clicking on the timer name in the user interface[4]:

- MicroWizard Fast Track P-Series: http://www.microwizard.com
- NewBold DT, TURBO, or DerbyStick: http://www.pinewood-derby-timer.com

Support for Other Timers and Gate Releases

The "Developers- Timer Messages" document describes the message protocols between derby-timer.jar and the server. Experimenters who build their own timer and/or gate release device may wish to implement this protocol and communicate directly with the DerbyNet server.

Alternatively, for home-brew or DIY timers, derby-timer.jar provides a -command argument to allow interacting with another program as if it were a serial port. E.g., for a python script that emulates a hardware timer, use something like:

```
java -jar derby-timer.jar -command python3 + -u + my-great-timer.py
```

---

2 There are apparently at least two versions of the Derby Magic firmware; they communicate at different baud rates. It's not known whether all versions of the Derby Magic timer can be automatically detected; accordingly, there are two manually-selectable entries on the list (one for each baud rate), in case the automatic detection process fails. If you find you have to use one of these, please get in touch.

3 If you have access to a SuperTimer II, any testing observations you can offer would be appreciated. Please get in touch!

4 Because these timers don't respond to probes from the host computer, the identity of the timer can't be confirmed until some recognizable communication from the timer occurs (e.g., a heat result is transmitted from the timer). The status of the timer will show as "UNCONFIRMED" until that happens.

Get in touch for further support for this scenario.

Automatic Gate Releases

Some timers support automatic release of the start gate under software control. These include:

- Derby Magic timer
- MiscJunk PDT timer
- MicroWizard Light Tree & Automatic Gate Release

Configuration for MicroWizard Light Tree & Automatic Gate Release

The MicroWizard Light Tree & Automatic Gate Release product interacts with the control of their laser start switch. Use the `fasttrack-automatic-gate-release` option if you're interfacing with one of these devices.

Without this option, management of the laser switch will cause the automatic gate release to open as soon as a new heat is queued up for racing.

If you're using derby-timer.jar, you can specify the flag on the command line:

```
java -jar derby-timer.jar -fasttrack-automatic-gate-release
```

Troubleshooting and Timer Log Files

If you experience difficulty with the connection to your timer, log files produced by the timer interface are an invaluable diagnostic tool.

Can't Run derby-timer.jar on macOS

macOS' malware protection scheme prevents running apps that aren't from an "identified" developer. An "identified" developer is one who pays Apple a $100-per-year developer registration fee. derby-timer.jar is likely to produce this message on a Mac.

Option 1: After you see this message, you can go to "System Preferences", and choose the "Security & Privacy" control panel. There, on the "General" tab, you should see an "Allow Anyway" option. Click that, and then you should be able to open derby-timer.jar

Here's a video showing the process.

Option 2: Follow the steps described in this Apple Support article.

Unable to connect to hosted instance: "ValidatorException"

If derby-timer.jar produces a Java exception about a ValidatorException while trying to connect to a cloud-hosted instance of DerbyNet, this is usually a consequence of using a version of Java older than Java 8 (Java 1.8). Upgrading to a more recent release fixes this problem. (The issue is that the Certificate Authority for the web site, Let's Encrypt, isn't included in the list of authorities trusted by older Java releases.)

Scanning Runs Without Detecting Timer

As described in the timer operation guide, a scan of the serial port(s) attempts to identify the attached race timer. For some timers, this is not possible, because the timer doesn't respond to interrogations from the host.

If your timer is one that can be identified, but the scan is not identifying it, check that the timer is plugged in and that it's actually connected to the computer. (In the chaos of setting up, it's easy to forget this.)

If that's not the problem, next consult the timer logs. (See “Getting Log Files,” below.) The log file shows all the data passing between derby-timer and the timer device. In particular, we'd want to know whether any traffic has been received from the device.

If no traffic is being received, suspect the cabling to the timer. (Also confirm what kind of timer it is, and check that it's expected to be identifiable.)

Please get in touch if you find that traffic is being passed but the timer is still not being identified.

Getting Log Files

Server-Side Timer Log Files

Both the in-browser timer interface and derby-timer.jar can send full log information to the DerbyNet server. Server-side timer logging is activated from the Timer Testing page.

Local Log Files With derby-timer.jar

derby-timer.jar writes a log file that records all its interactions with the timer or serial port, and with the DerbyNet server. For troubleshooting problems with a timer, the log file is the first place to look. The name of the log file always starts with “timer-” and ends with “.log”.

When derby-timer.jar is run from the command line, the log file is written to the current directory. When run by double-clicking, the log file is written to a temporary directory whose path depends on the platform.

Regardless where the derby-timer.jar log file is written, on all platforms its location can be determined directly from the user interface. The user interface window includes a menu bar with only one command, under the “File” menu: “Show Log File;” this command opens the directory containing the current log file.