
# cron

Cron V3 has been released!

To download the specific tagged release, run:

```
go get github.com/robfig/cron/v3@v3.0.0
```

Import it in your program as:

```
import "github.com/robfig/cron/v3"
```

It requires Go 1.11 or later due to usage of Go Modules.

Refer to the documentation here: http://godoc.org/github.com/robfig/cron

The rest of this document describes the the advances in v3 and a list of breaking changes for users that wish to upgrade from an earlier version.

## Upgrading to v3 (June 2019)

cron v3 is a major upgrade to the library that addresses all outstanding bugs, feature requests, and rough edges. It is based on a merge of master which contains various fixes to issues found over the years and the v2 branch which contains some backwards-incompatible features like the ability to remove cron jobs. In addition, v3 adds support for Go Modules, cleans up rough edges like the timezone support, and fixes a number of bugs.

New features:

- Support for Go modules. Callers must now import this library as `github.com/robfig/cron/v3`, instead of `gopkg.in/...`
- Fixed bugs:
  - 0f01e6b parser: fix combining of Dow and Dom (#70)
  - dbf3220 adjust times when rolling the clock forward to handle non-existent midnight (#157)
  - eeecf15 spec_test.go: ensure an error is returned on 0 increment (#144)
  - 70971dc cron.Entries(): update request for snapshot to include a reply channel (#97)
  - 1cba5e6 cron: fix: removing a job causes the next scheduled job to run too late (#206)
- Standard cron spec parsing by default (first field is "minute"), with an easy way to opt into the seconds field (quartz-compatible). Although, note that the year field (optional in Quartz) is not supported.
- Extensible, key/value logging via an interface that complies with the https://github.com/go-logr/logr project.
- The new Chain & JobWrapper types allow you to install "interceptors" to add cross-cutting behavior like the following:
  - Recover any panics from jobs
  - Delay a job's execution if the previous run hasn't completed yet
  - Skip a job's execution if the previous run hasn't completed yet

    - Log each job's invocations
    - Notification when jobs are completed

It is backwards incompatible with both v1 and v2. These updates are required:

- The v1 branch accepted an optional seconds field at the beginning of the cron spec. This is non-standard and has led to a lot of confusion. The new default parser conforms to the standard as described by the Cron wikipedia page.

  UPDATING: To retain the old behavior, construct your Cron with a custom parser:

  ```
  // Seconds field, required
  cron.New(cron.WithSeconds())

  // Seconds field, optional
  cron.New(
      cron.WithParser(
          cron.SecondOptional | cron.Minute | cron.Hour | cron.Dom | cron.Month |
  cron.Dow | cron.Descriptor))
  ```

- The Cron type now accepts functional options on construction rather than the previous ad-hoc behavior modification mechanisms (setting a field, calling a setter).

  UPDATING: Code that sets Cron.ErrorLogger or calls Cron.SetLocation must be updated to provide those values on construction.

- CRON_TZ is now the recommended way to specify the timezone of a single schedule, which is sanctioned by the specification. The legacy "TZ=" prefix will continue to be supported since it is unambiguous and easy to do so.

  UPDATING: No update is required.

- By default, cron will no longer recover panics in jobs that it runs. Recovering can be surprising (see issue #192) and seems to be at odds with typical behavior of libraries. Relatedly, the `cron.WithPanicLogger` option has been removed to accommodate the more general JobWrapper type.

  UPDATING: To opt into panic recovery and configure the panic logger:

  ```
  cron.New(cron.WithChain(
      cron.Recover(logger),  // or use cron.DefaultLogger
  ))
  ```

- In adding support for https://github.com/go-logr/logr, `cron.WithVerboseLogger` was removed, since it is duplicative with the leveled logging.

  UPDATING: Callers should use `WithLogger` and specify a logger that does not discard `Info` logs. For convenience, one is provided that wraps `*log.Logger`:

  ```
  cron.New(
      cron.WithLogger(cron.VerbosePrintfLogger(logger)))
  ```

### Background - Cron spec format

There are two cron spec formats in common usage:

- The "standard" cron format, described on the Cron wikipedia page and used by the cron Linux system utility.
- The cron format used by the Quartz Scheduler, commonly used for scheduled jobs in Java software

The original version of this package included an optional "seconds" field, which made it incompatible with both of these formats. Now, the "standard" format is the default format accepted, and the Quartz format is opt-in.