

---

# **Calends: Documented**

***Versión 0.1.0***

**Hennik Hunsaker**

**28 de febrero de 2023**



|  |            |
|--|------------|
| <b>1. Introduction</b>                                 | <b>3</b>   |
| <b>2. Features in Calends</b>                          | <b>5</b>   |
| <b>3. Installation of Calends</b>                      | <b>7</b>   |
| 3.1. Installing Calends for the Command Line . . . . . | 7          |
| 3.2. Installing Calends for Golang . . . . .           | 7          |
| 3.3. Installing Calends for C/C++ . . . . .            | 8          |
| 3.4. Installing Calends for Dart . . . . .             | 8          |
| 3.5. Installing Calends for JS/WASM . . . . .          | 8          |
| 3.6. Installing Calends for PHP . . . . .              | 9          |
| <b>4. Usage of Calends</b>                             | <b>11</b>  |
| 4.1. Using Calends from the Command Line . . . . .     | 11         |
| 4.2. Using Calends in Golang . . . . .                 | 20         |
| 4.3. Using Calends in C/C++ . . . . .                  | 30         |
| 4.4. Using Calends in Dart . . . . .                   | 41         |
| 4.5. Using Calends in JS/WASM . . . . .                | 51         |
| 4.6. Using Calends in PHP . . . . .                    | 60         |
| <b>5. Calendar Systems</b>                             | <b>71</b>  |
| 5.1. The Gregorian Calendar . . . . .                  | 71         |
| 5.2. Julian Day Count . . . . .                        | 72         |
| 5.3. Stardates . . . . .                               | 72         |
| 5.4. TAI64 Time . . . . .                              | 74         |
| 5.5. UNIX Time . . . . .                               | 75         |
| <b>6. Custom Calendars</b>                             | <b>77</b>  |
| 6.1. Custom Calendars in Golang . . . . .              | 77         |
| 6.2. Custom Calendars in C/C++ . . . . .               | 83         |
| 6.3. Custom Calendars in Dart . . . . .                | 89         |
| 6.4. Custom Calendars in JS/WASM . . . . .             | 94         |
| 6.5. Custom Calendars in PHP . . . . .                 | 98         |
| <b>7. Appendix</b>                                     | <b>103</b> |
| 7.1. Contributions . . . . .                           | 103        |

|                             |            |
|-----------------------------|------------|
| <b>Golang Package Index</b> | <b>105</b> |
| <b>PHP Namespace Index</b>  | <b>107</b> |
| <b>Índice</b>               | <b>109</b> |

|             |        |              |             |                      |               |      |           |           |     |     |
|-------------|--------|--------------|-------------|----------------------|---------------|------|-----------|-----------|-----|-----|
| LICENSE     | MIT    | MAIN DOCS    | PASSING     | GoDoc                | reference     | chat | on gitter | downloads | 331 |     |
| RELEASE     | V0.0.5 | RELEASE DATE | AUGUST 2018 | commits since v0.0.5 |               |      |           |           |     | 321 |
| last commit | today  |              |             |                      |               |      |           |           |     |     |
| maintained  | yes    | coverage     | 96%         | dependencies         | 8 out of date |      |           |           |     |     |

Calends is a library for handling dates and times across arbitrary calendar systems. It has a number of features essential to handling dates and times in any supported system, as well as the ability to add support for more systems easily, all without touching a line of its own code. Go ahead and read through the documentation here to learn more!



---

## Introduction

---

As mentioned before, Calends is a library for handling dates and times across arbitrary calendar systems. But what does that actually mean?

Let's say you're working on an application that uses dates. Pretty much anything can qualify, really – we use dates in everything, throughout our daily lives. Scheduling, journaling, historical research, projections into the future, or just displaying the current date in the UI. Now let's say you want your app to be used by people all over the globe. The current approach, used for decades, is to simply use the Gregorian Calendar, which has (partially as a side effect of this decision) become the default calendar system in use across the globe, for coordinating, tracking, and preserving events worldwide.

But this decision wasn't made with internationalization and localization in mind. It was made as a result of practicality, with limited computing capabilities at the time, and persists mostly as a result of laziness – if the entire world is already using it anyway, why bother with anything else? It has also persisted out of ignorance – many people aren't aware there are other calendars out there in the first place, never mind that several are still in active use to this day. Properly localizing applications should *include* adjusting the displayed date to use the preferred calendar system of the user.

Sadly, most of the solutions currently available for handling dates (and times) in software are purpose-built for a single calendar system, and use APIs entirely different from those meant to handle dates in others. This makes it very tricky to build an application that supports more than one calendar system at a time. Each new calendar system requires hours of work to learn, connect, and usually abstract to a point where it is usable within the larger application, and even that's no guarantee the values can be stored or compared accurately.

That's what Calends set out to solve. It provides a single interface for interacting with any supported calendar system, and an easy way to extend it to support others, so that once you've added support for a calendar system once, you have that support anywhere, without having to rewrite anything to fit your next application. Additionally, you can take full advantage of any calendar system implemented by anybody else.

Accept date/time input in any calendar system, perform date/time calculations on the resulting value, again in any calendar system, and then display the result – yes, in any calendar system. Dates are stored, internally, in an extremely accurate value that can track dates out 146 *billion* years into the past or future, with a resolution of  $10^{-45}$  seconds, which is smaller than Planck Time<sup>1</sup>. In other words, it should be more than sufficient to record instants of any duration and

---

<sup>1</sup> Planck Time is the smallest meaningful unit of time, and is about  $54 \times 10^{-45}$  seconds. It corresponds to the amount of time it takes a photon (traveling at the speed of light through a vacuum) to traverse one Planck Length, which itself is about  $10^{-20}$  times the diameter of a proton. Even quantum interactions below this scale lose any meaning, and so values below them are considered extraneous, in addition to being entirely unmeasurable with current technologies and techniques.

resolution desired for any conceived use case.

---

## Features in Calends

---

For a current indication of which of these features are fully implemented at the moment, check [the README](#).

- **Large range and high precision** Calends understands dates  $2^{62}$  seconds into the future or past, in units as small as  $10^{-45}$  seconds – that’s over 146 billion years into the past or future (146 138 512 313 years, 169 days, 10 hours, 5 minutes, and 28 seconds from CE 1970 Jan 01 00:00:00 TAI Gregorian), at resolutions smaller than Planck Time ( $54 \times 10^{-45}$  seconds, and the smallest meaningful duration even on the quantum scale). That encompasses well beyond the expected lifespan of the Universe, at resolutions enough to represent quantum events.
- **Supports date (and time) values in multiple calendar systems** Supported out of the box are the following (all systems are proleptic – extrapolated beyond the officially-defined limits – unless specified otherwise):
  - **Unix time** A count of the number of seconds since CE 1970 Jan 01 00:00:00 UTC Gregorian
  - **TAI64** Essentially Unix time plus  $2^{62}$ , but using TAI seconds instead of UTC seconds, so times can be converted unambiguously (UTC uses leap seconds to keep the solar zenith at noon, while TAI is a simple, unadjusted count). Calends supports an extended version of this spec, with three more components, to encode out to 45 places instead of just 18; this is also actually the internal time scale used by Calends itself, which is how it can support such a broad range of dates at such a high resolution.
    - Automatic calculation of leap second offsets
    - Estimation of undefined past and future leap second insertions
    - Automatic updates for handling leap second insertions
  - **Gregorian** The current international standard calendar system
    - Disconnect from native `time.Time` implementation, and its limitations
  - **Julian** The previous version of the Gregorian calendar

- *Julian Day Count* A count of days since BCE 4713 Jan 01 12:00:00 UTC Julian (proleptic)
  - **Hebrew**
  - **Persian**
  - **Chinese** Several variants
  - **Meso-American** Commonly called Mayan, but used by several cultures in the region
  - **Discordian**
  - *Stardate* Yes, the ones from *Star Trek*<sup>TM</sup>; several variants exist
- **Encodes both time spans and instants in a single interface** The library treats the time values it encodes as [start, end) sets (that is, the start point is included in the range, as is every point between start and end, but the end point itself is `_not_` included in the range). This allows duration to accurately be `end - start` in all cases. (And yes, that also means you can create spans with `duration < 0`.)
  - **Supports calculations and comparisons on spans and instants** Addition, subtraction, intersection, combination, gap calculation, overlap detection, and similar operations are all supported directly on Calends values.
  - **Conversion to/from native date/time types** While this is possible by using a string representation as an intermediary, in either direction, some data and precision is lost in such a conversion. Instead, Calends supports conversion to and from such types directly, preserving as much data and accuracy as each native type provides.
  - **Geo-temporally aware** The library provides methods for passing a location instead of a calendar system, and selecting an appropriate calendar based on which was most common in that location at that point in time. *(Some guess work is involved in this process when parsing dates, so it is still preferred to supply the calendar system, if known, when parsing.)*
  - **Time zone support**
  - **Well-defined interfaces for extending the library** Add more calendar systems, type conversions, or geo-temporal relationships without forking/modifying the library itself.
  - **Shared library (.so/.dll/.dylib)** In order to use the library outside of Golang projects, we first need to export its functionality in a shared library, which can then be accessed from other programming environments and applications, generally via FFI.
  - **WebAssembly binary** In order to use the library in the browser, we first need to export its functionality in a WebAssembly (WASM) binary, which can then be accessed by JavaScript. (Go currently doesn't support the WASI standard, so the functions are registered into the global namespace rather than being exported by WebAssembly itself. More on that in the JS docs.)

---

## Installation of Calends

---

The steps here vary based on which programming language(s) you're using. Golang is a simple install and import. Other languages use a language-specific wrapper around the compiled shared library. Select your language to take a deeper look.

### 3.1 Installing Calends for the Command Line

You can grab a calends binary for your platform OS/architecture from the [GitHub Releases](#) page, and just run it directly. Alternately, you can clone the source and build it from there:

```
# Sample Linux steps:
mkdir -p $GOPATH/src/github.com/danhunsaker
cd $GOPATH/src/github.com/danhunsaker
git clone https://github.com/danhunsaker/calends
cd calends
go get ./...
go build -o calends ./cli
```

### 3.2 Installing Calends for Golang

Install the library with `go get github.com/danhunsaker/calends`, and then import `"github.com/danhunsaker/calends"` wherever you intend to use it. If you're using another Go dependency manager, you'll need to use its dependency installation method instead; consult its documentation for more details, as we can't possibly cover them all here.

## 3.3 Installing Calends for C/C++

### 3.3.1 Binary Install

For use with C/C++, simply grab the latest version of `libcalends` from the [GitHub Releases page](#), and extract its contents wherever your compiler expects to find `.h` and `.so/.dll/.dylib` files. Be sure to grab the correct version for your architecture!

### 3.3.2 Source Install

To install from source, you'll need Go 1.9+ installed to use its compiler. Clone the repository, build `libcalends`, then copy the resulting `.so/.dll/.dylib` and `.h` files to wherever your C/C++ compiler expects to find them.

```
# Sample Linux steps:
mkdir -p $GOPATH/src/github.com/danhunsaker
cd $GOPATH/src/github.com/danhunsaker
git clone https://github.com/danhunsaker/calends
cd weekends/libcalends
go get ../...
go build -v -i -buildmode=c-shared -o libcalends.so
```

Adjust the above example commands as needed for your actual development OS.

## 3.4 Installing Calends for Dart

For use with Dart, use `pub` as normal:

```
dart pub add weekends
```

Or with Flutter:

```
flutter pub add weekends
```

You'll also need to grab the latest version of `libcalends` from the [GitHub Releases page](#), and extract its contents (you can skip any `.h` files) into your project's root directory, next to `pubspec.yaml`. Be sure to grab the correct archive for your architecture!

## 3.5 Installing Calends for JS/WASM

For use with JS, use `npm` (or your preferred package manager):

```
.. code-block:: bash

npm install -s weekends
```

This will pull in the JS wrapper package as well as the corresponding WASM binary.

For use on the server, that's pretty much it. The library takes care of the rest.

For use on the web, you'll need to ensure the WASM is accessible to the server, next to the library itself. The easiest way to ensure this is to pull in `calends.js` directly via `<script>` tag, but if you use a package to compile/minify/etc

your JS dependencies, you'll need to configure that package to include `calends.wasm` alongside your script(s). Here's an example for webpack:

```
.. code-block:: javascript

const CopyPlugin = require("copy-webpack-plugin");

// ...

module.exports = {
  // ...
  plugins: [
    new CopyPlugin({
      patterns: [
        { from: "node_modules/calends/calends.wasm",
          to: "[name][ext]" },
      ],
    }),
  ],
};
```

It's not clean, but until Go compiles compliant WASM binaries, it's the best we can do right now, since we can't use *WebAssembly ESM Integration* <<https://github.com/WebAssembly/esm-integration/tree/main/proposals/esm-integration>> yet. Once it defines `exports` correctly, we can drop much of the JS wrapper and focus purely on translating bare functions into full classes exclusively.

## 3.6 Installing Calends for PHP

For use with PHP, use Composer to install the PHP FFI wrapper:

```
composer install danhunsaker/calends
```

The post-install script will grab the appropriate `libcalends` for your system, along with the relevant header file. From there, simply update your `php.ini` to load the FFI extension (if not already loaded) and preload the header file:

```
extension=ffi.so
ffi.preload=/path/to/your/code/vendor/lib/calends-phpffi.h
```

If you don't have access to edit your `php.ini`, ensure the FFI extension is available and enabled, then manually load the header file in your code:

```
FFI::load(__DIR__ . "/vendor/lib/calends-phpffi.h");
```



---

## Usage of Calends

---

Once you have *installed Calends*, you'll want to know how to actually make use of it in your own projects. The exact approach for this varies by language, so we've broken it into multiple sections to make it easier to wade through and find what you need. Find your language, below, and dig in a bit further to see how to do everything you need!

### 4.1 Using Calends from the Command Line

Calends can be used from the command line directly, though some of its features are limited or unavailable. Specifically, it doesn't support custom calendars, so you'll need to ensure you build it with the calendar you want already loaded.

#### 4.1.1 Command Line Options

The available options for `calends`, on the command line directly, are the following:

**convert** <from-calendar> <from-format> <to-calendar> <to-format> [<date>]

- **from-calendar** The calendar system to parse the date/time with.
- **from-format** The format the date/time is expected to use.
- **to-calendar** The calendar system to format the date/time with.
- **to-format** The format the date/time is expected to use.
- **date** The value to convert.

Converts a date from one calendar/format to another. If `date` isn't provided in the arguments, it's read from `/dev/stdin` instead.

**parse** <from-calendar> <from-format> [<date>]

- **from-calendar** The calendar system to parse the date/time with.
- **from-format** The format the date/time is expected to use.
- **date** The value to parse.

Converts a date from the given calendar/format to a portable/unambiguous date stamp. The output from this command can then be used as input to others.

**format** <to-calendar> <to-format> [<stamp>]

- **to-calendar** The calendar system to format the date/time with.
- **to-format** The format the date/time is expected to use.
- **stamp** The value to format.

Converts a date stamp from the *parse* command to the given calendar/format.

**offset** <offset-calendar> [<offset> [<stamp>]]

- **offset-calendar** The calendar system to interpret the offset with.
- **offset** The offset to add.
- **stamp** The value to add the offset to.

Adds an offset to the date stamp from the *parse* command.

There is also a `calends compare`, whose options are these:

**contains** [<stamp1> [<stamp2>]]

- **stamp1** The value to compare.
- **stamp2** The value to compare the other to.

Compares stamp1 to stamp2, and returns whether stamp1 contains stamp2.

**overlaps** [<stamp1> [<stamp2>]]

- **stamp1** The value to compare.
- **stamp2** The value to compare the other to.

Compares stamp1 to stamp2, and returns whether stamp1 overlaps with stamp2.

**abuts** [<stamp1> [<stamp2>]]

- **stamp1** The value to compare.
- **stamp2** The value to compare the other to.

Compares stamp1 to stamp2, and returns whether stamp1 abuts stamp2.

**same** [<stamp1> [<stamp2>]]

- **stamp1** The value to compare.
- **stamp2** The value to compare the other to.

Compares stamp1 to stamp2, and returns whether stamp1 is the same as stamp2.

**shorter** [<stamp1> [<stamp2>]]

- **stamp1** The value to compare.
- **stamp2** The value to compare the other to.

Compares stamp1 to stamp2, and returns whether stamp1 is shorter than stamp2.

**same-duration** [<stamp1> [<stamp2>]]

- **stamp1** The value to compare.
- **stamp2** The value to compare the other to.

Compares stamp1 to stamp2, and returns whether stamp1 is the same duration as stamp2.

**longer** [<stamp1> [<stamp2>]]

- **stamp1** The value to compare.
- **stamp2** The value to compare the other to.

Compares stamp1 to stamp2, and returns whether stamp1 is longer than stamp2.

**before** [<stamp1> [<stamp2>]]

- **stamp1** The value to compare.
- **stamp2** The value to compare the other to.

Compares stamp1 to stamp2, and returns whether stamp1 is before stamp2.

**start-before** [<stamp1> [<stamp2>]]

- **stamp1** The value to compare.
- **stamp2** The value to compare the other to.

Compares stamp1 to stamp2, and returns whether stamp1 starts before stamp2.

**end-before** [<stamp1> [<stamp2>]]

- **stamp1** The value to compare.
- **stamp2** The value to compare the other to.

Compares stamp1 to stamp2, and returns whether stamp1 ends before stamp2.

**during** [<stamp1> [<stamp2>]]

- **stamp1** The value to compare.
- **stamp2** The value to compare the other to.

Compares stamp1 to stamp2, and returns whether stamp1 is during stamp2.

**start-during** [<stamp1> [<stamp2>]]

- **stamp1** The value to compare.
- **stamp2** The value to compare the other to.

Compares stamp1 to stamp2, and returns whether stamp1 starts during stamp2.

**end-during** [<stamp1> [<stamp2>]]

- **stamp1** The value to compare.
- **stamp2** The value to compare the other to.

Compares stamp1 to stamp2, and returns whether stamp1 ends during stamp2.

**after** [<stamp1> [<stamp2>]]

- **stamp1** The value to compare.
- **stamp2** The value to compare the other to.

Compares stamp1 to stamp2, and returns whether stamp1 is after stamp2.

**start-after** [<stamp1> [<stamp2>]]

- **stamp1** The value to compare.
- **stamp2** The value to compare the other to.

Compares stamp1 to stamp2, and returns whether stamp1 starts after stamp2.

**end-after** [<stamp1> [<stamp2>]]

- **stamp1** The value to compare.
- **stamp2** The value to compare the other to.

Compares stamp1 to stamp2, and returns whether stamp1 ends after stamp2.

## 4.1.2 Interactive/Batch Mode

### Create

**parse** <calendar> <format> <date> <target>

- **calendar** The calendar system to parse the date/time with.
- **format** The format the date/time is expected to use.
- **date** The value to parse.
- **target** The name to give the result.

Creates a new Calends value, using **calendar** to select a calendar system, and **format** to describe the contents of **date** to parse. The result is stored as **target**, so it can be used later on by other commands.

**parse-range** <calendar> <format> <date> <end-date> <target>

- **calendar** The calendar system to parse the dates/times with.
- **format** The format the dates/times are expected to use.
- **date** The start date to parse.
- **end-date** The end date to parse.
- **target** The name to give the result.

Creates a new Calends value, using **calendar** to select a calendar system, and **format** to describe the contents of **date** and **end-date** to parse. The result is stored as **target**, so it can be used later on by other commands.

## Read

**date** <calendar> <format> <source>

- **calendar** The calendar system to format the date/time with.
- **format** The format the date/time is expected to be in.
- **source** The name of the Calends value to use.

Retrieves the start date of **source** as a string. The value is generated by the calendar system given in **calendar**, according to the format string in **format**.

**end-date** <calendar> <format> <source>

- **calendar** The calendar system to format the date/time with.
- **format** The format the date/time is expected to be in.
- **source** The name of the Calends value to use.

Retrieves the end date of **source** as a string. The value is generated by the calendar system given in **calendar**, according to the format string in **format**.

## Update

**set-date** <calendar> <format> <date> <source> <target>

- **calendar** The calendar system to parse the date/time with.
- **format** The format the date/time is expected to use.
- **date** The value to parse the date/time from.
- **source** The name of the Calends value to use.
- **target** The name to give the result.

Sets the start date of **target** based on **source**'s current value. The inputs are the same as for [parse](#), above.

**set-end-date** <calendar> <format> <date> <source> <target>

- **calendar** The calendar system to parse the date/time with.
- **format** The format the date/time is expected to use.
- **date** The value to parse the date/time from.
- **source** The name of the Calends value to use.
- **target** The name to give the result.

Sets the end date of **target** based on **source**'s current value. The inputs are the same as for [parse](#), above.

## Manipulate

**add** <calendar> <offset> <source> <target>

- **calendar** The calendar system to parse the offset with.
- **offset** The value to parse the offset from.
- **source** The name of the Calends value to use.
- **target** The name to give the result.

Increases the end date of **source**'s current value by **offset**, as interpreted by the calendar system given in **calendar**.

**add-from-end** <calendar> <offset> <source> <target>

- **calendar** The calendar system to parse the offset with.
- **offset** The value to parse the offset from.
- **source** The name of the Calends value to use.
- **target** The name to give the result.

Increases the start date of **source**'s current value by **offset**, as interpreted by the calendar system given in **calendar**.

**subtract** <calendar> <offset> <source> <target>

- **calendar** The calendar system to parse the offset with.
- **offset** The value to parse the offset from.
- **source** The name of the Calends value to use.
- **target** The name to give the result.

Works the same as [add](#), except it decreases the start date, rather than increasing it.

**subtract-from-end** <calendar> <offset> <source> <target>

- **calendar** The calendar system to parse the offset with.
- **offset** The value to parse the offset from.
- **source** The name of the Calends value to use.
- **target** The name to give the result.

Works the same as [add-from-end](#), except it decreases the end date, rather than increasing it.

**next** <calendar> <offset> <source> <target>

- **calendar** The calendar system to parse the offset with.
- **offset** The value to parse the offset from.
- **source** The name of the Calends value to use.
- **target** The name to give the result.

Sets **target** to a Calends value of **offset** duration (as interpreted by the calendar system given in **calendar**), which abuts the end of **source**. If **offset** is empty, **calendar** is ignored, and **source**'s duration is used instead.

**previous** <calendar> <offset> <source> <target>

- **calendar** The calendar system to parse the offset with.
- **offset** The value to parse the offset from.
- **source** The name of the Calends value to use.
- **target** The name to give the result.

Sets **target** to a Calends value of **offset** duration (as interpreted by the calendar system given in **calendar**), which abuts the start of **source**. If **offset** is empty, **calendar** is ignored, and **source**'s duration is used instead.

## Combine

**merge** <source> <combine> <target>

- **source** The name of the Calends value to use.
- **combine** The Calends value to merge.
- **target** The name to give the result.

Sets **target** to a value spanning from the earliest start date to the latest end date between **source** and **combine**.

**intersect** <source> <combine> <target>

- **source** The name of the Calends value to use.
- **combine** The Calends value to intersect.
- **target** The name to give the result.

Sets **target** to a value spanning the overlap between **source** and **combine**. If **source** and **combine** don't overlap, returns an error.

**gap** <source> <combine> <target>

- **source** The name of the Calends value to use.
- **combine** The Calends value to gap.
- **target** The name to give the result.

Sets **target** to a value spanning the gap between **source** and **combine**. If **source** and **combine** overlap (and there is, therefore, no gap to return), returns an error.

## Compare

**difference** <source> <compare> <mode>

- **source** The name of the Calends value to use.
- **compare** The Calends value to compare.
- **mode** The comparison mode.

Returns the difference of **source** minus **compare**, using **mode** to select which values to use in the calculation. Valid modes include:

- **start** - use the start date of both **source** and **compare**
- **duration** - use the duration of both **source** and **compare**

- **end** - use the end date of both **source** and **compare**
- **start-end** - use the start of **source**, and the end of **compare**
- **end-start** - use the end of **source**, and the start of **compare**

**compare** <source> <compare> <mode>

- **source** The name of the Calends value to use.
- **compare** The Calends value to compare.
- **mode** The comparison mode.

Returns -1 if **source** is less than **compare**, 0 if they are equal, and 1 if **source** is greater than **compare**, using **mode** to select which values to use in the comparison. Valid modes are the same as for *difference*, above.

**contains** <source> <compare>

- **source** The name of the Calends value to use.
- **compare** The Calends value to compare.

Checks whether **source** contains all of **compare**.

**overlaps** <source> <compare>

- **source** The name of the Calends value to use.
- **compare** The Calends value to compare.

Checks whether **source** overlaps with **compare**.

**abuts** <source> <compare>

- **source** The name of the Calends value to use.
- **compare** The Calends value to compare.

Checks whether **source** abuts **compare** (that is, whether one begins at the same instant the other ends).

**is-same** <source> <compare>

- **source** The name of the Calends value to use.
- **compare** The Calends value to compare.

Checks whether **source** covers the same span of time as **compare**.

**is-shorter** <source> <compare>

- **source** The name of the Calends value to use.
- **compare** The Calends value to compare.

Compares the duration of **source** and **compare**.

**is-same-duration** <source> <compare>

- **source** The name of the Calends value to use.
- **compare** The Calends value to compare.

Compares the duration of **source** and **compare**.

**is-longer** <source> <compare>

- **source** The name of the Calends value to use.
- **compare** The Calends value to compare.

Compares the duration of **source** and **compare**.

**is-before** <source> <compare>

- **source** The name of the Calends value to use.
- **compare** The Calends value to compare.

Compares the entirety of **source** with the start date of **compare**.

**starts-before** <source> <compare>

- **source** The name of the Calends value to use.
- **compare** The Calends value to compare.

Compares the start date of **source** with the start date of **compare**.

**ends-before** <source> <compare>

- **source** The name of the Calends value to use.
- **compare** The Calends value to compare.

Compares the end date of **source** with the start date of **compare**.

**is-during** <source> <compare>

- **source** The name of the Calends value to use.
- **compare** The Calends value to compare.

Checks whether the entirety of **source** lies between the start and end dates of **compare**.

**starts-during** <source> <compare>

- **source** The name of the Calends value to use.
- **compare** The Calends value to compare.

Checks whether the start date of **source** lies between the start and end dates of **compare**.

**ends-during** <source> <compare>

- **source** The name of the Calends value to use.
- **compare** The Calends value to compare.

Checks whether the end date of **source** lies between the start and end dates of **compare**.

**is-after** <source> <compare>

- **source** The name of the Calends value to use.
- **compare** The Calends value to compare.

Compares the entirety of **source** with the end date of **compare**.

**starts-after** <source> <compare>

- **source** The name of the Calends value to use.
- **compare** The Calends value to compare.

Compares the start date of **source** with the end date of **compare**.

**ends-after** <source> <compare>

- **source** The name of the Calends value to use.
- **compare** The Calends value to compare.

Compares the end date of **source** with the end date of **compare**.

## 4.2 Using Calends in Golang

Calends exposes a very small handful of things for use outside the library itself. One is the [Calends](#) class, documented here, which should be the only interface users of the library ever need to touch.

Another thing Calends exposes is the `calendars.TAI64NARUXTIME` class, used to store and manipulate the instants of time which make up a [Calends](#) instance. The rest are interfaces for extending the library's functionality. These are covered in the [Custom Calendars in Golang](#) section.

---

**Nota:** [Calends](#) objects are immutable - all methods return a new [Calends](#) object where they might otherwise alter the current one. This is true even of the `Calends.Set*` methods. This has the side effect of using more memory to perform manipulations than updating values on an existing object would. It makes many operations safer, though, than mutable objects would allow.

---

### Calends

The main entry point to Calends and its functionality. [Calends](#) objects should only be created with the [Create](#) function, and never directly (especially given its values are all unexported ones).

#### 4.2.1 Create

func **Create**(value interface, calendar string, format string) (Calends, error)

##### Parámetros

- **value** (interface{}) – The value to parse the date(s)/time(s) from.
- **calendar** (string) – The calendar system to parse the date(s)/time(s) with.
- **format** (string) – The format the date(s)/time(s) is/are expected to use.

**Devuelve** A new [Calends](#) object

**Tipo del valor devuelto** [Calends](#)

**Devuelve** error when an error occurs; nil otherwise

**Tipo del valor devuelto** error or nil

Creates a new [Calends](#) object, using **calendar** to select a calendar system, and **format** to parse the contents of **value** into the [Calends](#) object's internal instants. The type of **value** can vary based on the calendar system itself, but generally speaking can always be a string.

In any case, the value can always be a `map[string]interface{}`, where the keys are any two of `start`, `end`, and `duration`. If all three are provided, `duration` is ignored in favor of calculating it directly. If only one of the listed keys is provided, `value` is passed to the calendar system itself unchanged.

The calendar system then converts `value` to a `calendars.TAI64NARUXTIME` instant, which the `Calends` object sets to the appropriate internal value.

## 4.2.2 Read

```
func (Calends) Date(calendar string, format string) (string, error)
```

### Parámetros

- **calendar** (string) – The calendar system to format the date/time with.
- **format** (string) – The format the date/time is expected to be in.

**Devuelve** The start date of the `Calends` object

**Tipo del valor devuelto** string

**Devuelve** error when an error occurs; nil otherwise

**Tipo del valor devuelto** error or nil

Retrieves the start date of the `Calends` object as a string. The value is generated by the calendar system given in `calendar`, according to the format string in `format`.

```
func (Calends) EndDate(calendar string, format string) (string, error)
```

### Parámetros

- **calendar** (string) – The calendar system to format the date/time with.
- **format** (string) – The format the date/time is expected to be in.

**Devuelve** The end date of the `Calends` object

**Tipo del valor devuelto** string

**Devuelve** error when an error occurs; nil otherwise

**Tipo del valor devuelto** error or nil

Retrieves the end date of the `Calends` object as a string. The value is generated by the calendar system given in `calendar`, according to the format string in `format`.

```
func (Calends) Duration() → Float
```

**Devuelve** The duration of the `Calends` object

**Tipo del valor devuelto** math/big.(\*Float)

Retrieves the duration of the `Calends` object as an arbitrary-precision floating point number. This value will be 0 if the `Calends` object is an instant.

### 4.2.3 Update

func (Calends) **SetDate**(stamp interface, calendar string, format string) (Calends, error)

**Parámetros**

- **value** (interface{}) – The value to parse the date/time from.
- **calendar** (string) – The calendar system to parse the date/time with.
- **format** (string) – The format the date/time is expected to use.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Devuelve** error when an error occurs; nil otherwise

**Tipo del valor devuelto** error or nil

Sets the start date of a *Calends* object, based on the *Calends* object's current value. The inputs are the same as for *Create*, above, except the string → value map option isn't available, since you're already specifically setting the start value explicitly.

func (Calends) **SetEndDate**(stamp interface, calendar string, format string) (Calends, error)

**Parámetros**

- **value** (interface{}) – The value to parse the date/time from.
- **calendar** (string) – The calendar system to parse the date/time with.
- **format** (string) – The format the date/time is expected to use.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Devuelve** error when an error occurs; nil otherwise

**Tipo del valor devuelto** error or nil

Sets the end date of a *Calends* object, based on the *Calends* object's current value. The inputs are the same as for *Create*, above, except the string → value map option isn't available, since you're already specifically setting the end value explicitly.

func (Calends) **SetDuration**(duration interface, calendar string) (Calends, error)

**Parámetros**

- **duration** (interface{}) – The value to parse the new duration from.
- **calendar** (string) – The calendar system to parse the date/time with.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Devuelve** error when an error occurs; nil otherwise

**Tipo del valor devuelto** error or nil

Sets the duration of a *Calends* object, by adjusting its end point, and using the current start point as an anchor. The duration value is interpreted by the calendar system given in *calendar*, so is subject to any of its rules.

func (Calends) SetDurationFromEnd(*duration* interface, *calendar* string) (Calends, error)

**Parámetros**

- **duration** (interface{}) – The value to parse the new duration from.
- **calendar** (string) – The calendar system to parse the date/time with.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Devuelve** error when an error occurs; nil otherwise

**Tipo del valor devuelto** error or nil

Sets the duration of a *Calends* object, by adjusting its start point, and using the current end point as an anchor. The duration value is interpreted by the calendar system given in *calendar*, so is subject to any of its rules.

## 4.2.4 Manipulate

func (Calends) Add(*offset* interface, *calendar* string) (Calends, error)

**Parámetros**

- **offset** (interface{}) – The value to parse the offset from.
- **calendar** (string) – The calendar system to parse the date/time with.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Devuelve** error when an error occurs; nil otherwise

**Tipo del valor devuelto** error or nil

Increases the end date of the *Calends* object's current value by *offset*, as interpreted by the calendar system given in *calendar*.

func (Calends) AddFromEnd(*offset* interface, *calendar* string) (Calends, error)

**Parámetros**

- **offset** (interface{}) – The value to parse the offset from.
- **calendar** (string) – The calendar system to parse the date/time with.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Devuelve** error when an error occurs; nil otherwise

**Tipo del valor devuelto** error or nil

Increases the start date of the *Calends* object's current value by *offset*, as interpreted by the calendar system given in *calendar*.

func (Calends) Subtract(*offset* interface, *calendar* string) (Calends, error)

**Parámetros**

- **offset** (interface{}) – The value to parse the offset from.
- **calendar** (string) – The calendar system to parse the date/time with.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Devuelve** error when an error occurs; nil otherwise

**Tipo del valor devuelto** error or nil

Works the same as Add, except it decreases the start date, rather than increasing it.

```
func (Calends) SubtractFromEnd(offset interface, calendar string) (Calends, error)
```

**Parámetros**

- **offset** (interface{}) – The value to parse the offset from.
- **calendar** (string) – The calendar system to parse the date/time with.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Devuelve** error when an error occurs; nil otherwise

**Tipo del valor devuelto** error or nil

Works the same as AddFromEnd, except it decreases the end date, rather than increasing it.

```
func (Calends) Next(offset interface, calendar string) (Calends, error)
```

**Parámetros**

- **offset** (interface{}) – The value to parse the offset from.
- **calendar** (string) – The calendar system to parse the date/time with.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Devuelve** error when an error occurs; nil otherwise

**Tipo del valor devuelto** error or nil

Returns a *Calends* object of offset duration (as interpreted by the calendar system given in calendar), which abuts the *Calends* object's current value. If offset is empty, calendar is ignored, and the current object's duration is used instead.

```
func (Calends) Previous(offset interface, calendar string) (Calends, error)
```

**Parámetros**

- **offset** (interface{}) – The value to parse the offset from.
- **calendar** (string) – The calendar system to parse the date/time with.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Devuelve** error when an error occurs; nil otherwise

**Tipo del valor devuelto** error or nil

Returns a *Calends* object of offset duration (as interpreted by the calendar system given in calendar), which abuts the *Calends* object's current value. If offset is empty, calendar is ignored, and the current object's duration is used instead.

### 4.2.5 Combine

func (Calends) Merge(c2 Calends) (Calends, error)

**Parámetros**

- **c2** (Calends) – The Calends object to merge.

**Devuelve** A new Calends object

**Tipo del valor devuelto** Calends

**Devuelve** error when an error occurs; nil otherwise

**Tipo del valor devuelto** error or nil

Returns a Calends object spanning from the earliest start date to the latest end date between the current Calends object and c2.

func (Calends) Intersect(c2 Calends) (Calends, error)

**Parámetros**

- **c2** (Calends) – The Calends object to intersect.

**Devuelve** A new Calends object

**Tipo del valor devuelto** Calends

**Devuelve** error when an error occurs; nil otherwise

**Tipo del valor devuelto** error or nil

Returns a Calends object spanning the overlap between the current Calends object and c2. If the current object and c2 don't overlap, returns an error.

func (Calends) Gap(c2 Calends) (Calends, error)

**Parámetros**

- **c2** (Calends) – The Calends object to gap.

**Devuelve** A new Calends object

**Tipo del valor devuelto** Calends

**Devuelve** error when an error occurs; nil otherwise

**Tipo del valor devuelto** error or nil

Returns a Calends object spanning the gap between the current Calends object and c2. If the current object and c2 overlap (and there is, therefore, no gap to return), returns an error.

### 4.2.6 Compare

func (Calends) Difference(c2 Calends, mode string) → Float

**Parámetros**

- **c2** (Calends) – The Calends object to compare.
- **mode** (string) – The comparison mode.

**Devuelve** The difference, as an arbitrary-precision floating point number

**Tipo del valor devuelto** math/big.Float

Returns the difference of the current *Calends* object minus *c2*, using *mode* to select which values to use in the calculation. Valid modes include:

- **start** - use the start date of both the current object and *c2*
- **duration** - use the duration of both the current object and *c2*
- **end** - use the end date of both the current object and *c2*
- **start-end** - use the start of the current object, and the end of *c2*
- **end-start** - use the end of the current object, and the start of *c2*

func (**Calends**) **Compare**(*c2* Calends, *mode* string) → int

**Parámetros**

- **c2** (*Calends*) – The *Calends* object to compare.
- **mode** (string) – The comparison mode.

**Devuelve** A standard comparison result

**Tipo del valor devuelto** int

Returns -1 if the current *Calends* object is less than *c2*, 0 if they are equal, and 1 if the current object is greater than *c2*, using *mode* to select which values to use in the comparison. Valid modes are the same as for (*Calends*) *Difference*, above.

func (**Calends**) **Contains**(*c2* Calends) → bool

**Parámetros**

- **c2** (*Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Checks whether the current *Calends* object contains all of *c2*.

func (**Calends**) **Overlaps**(*c2* Calends) → bool

**Parámetros**

- **c2** (*Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Checks whether the current *Calends* object overlaps with *c2*.

func (**Calends**) **Abuts**(*c2* Calends) → bool

**Parámetros**

- **c2** (*Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Checks whether the current *Calends* object abuts *c2* (that is, whether one begins at the same instant the other ends).

```
func (Calends) IsSame(c2 Calends) → bool
```

**Parámetros**

- **c2** (*Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Checks whether the current *Calends* object covers the same span of time as c2.

```
func (Calends) IsShorter(c2 Calends) → bool
```

**Parámetros**

- **c2** (*Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the duration of the current *Calends* object and c2.

```
func (Calends) IsSameDuration(c2 Calends) → bool
```

**Parámetros**

- **c2** (*Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the duration of the current *Calends* object and c2.

```
func (Calends) IsLonger(c2 Calends) → bool
```

**Parámetros**

- **c2** (*Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the duration of the current *Calends* object and c2.

```
func (Calends) IsBefore(c2 Calends) → bool
```

**Parámetros**

- **c2** (*Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the entirety of the current *Calends* object with the start date of c2.

```
func (Calends) StartsBefore(c2 Calends) → bool
```

**Parámetros**

- **c2** (*Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the start date of the current *Calends* object with the start date of *c2*.

```
func (Calends) EndsBefore(c2 Calends) → bool
```

**Parámetros**

- *c2* (*Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the end date of the current *Calends* object with the start date of *c2*.

```
func (Calends) IsDuring(c2 Calends) → bool
```

**Parámetros**

- *c2* (*Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Checks whether the entirety of the current *Calends* object lies between the start and end dates of *c2*.

```
func (Calends) StartsDuring(c2 Calends) → bool
```

**Parámetros**

- *c2* (*Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Checks whether the start date of the current *Calends* object lies between the start and end dates of *c2*.

```
func (Calends) EndsDuring(c2 Calends) → bool
```

**Parámetros**

- *c2* (*Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Checks whether the end date of the current *Calends* object lies between the start and end dates of *c2*.

```
func (Calends) IsAfter(c2 Calends) → bool
```

**Parámetros**

- *c2* (*Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the entirety of the current *Calends* object with the end date of *c2*.

```
func (Calends) StartsAfter(c2 Calends) → bool
```

**Parámetros**

- *c2* (*Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the start date of the current *Calends* object with the end date of c2.

```
func (Calends) EndsAfter(c2 Calends) → bool
```

**Parámetros**

- **c2** (*Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the end date of the current *Calends* object with the end date of c2.

## 4.2.7 Export

```
func (Calends) String() → string
```

**Devuelve** The string representation of the current value.

**Tipo del valor devuelto** string

Implements the `fmt.Stringer` interface.

```
func (Calends) MarshalText() ([]byte, error)
```

**Devuelve** A byte slice containing the marshalled text.

**Tipo del valor devuelto** []byte

**Devuelve** Any error that occurs.

**Tipo del valor devuelto** error

Implements the `encoding.TextMarshaler` interface.

```
func (*Calends) UnmarshalText(in []byte) → error
```

**Parámetros**

- **in** ([]byte) – A byte slice containing the marshalled text.

**Devuelve** Any error that occurs.

**Tipo del valor devuelto** error

Implements the `encoding.TextUnmarshaler` interface.

```
func (Calends) MarshalJSON() ([]byte, error)
```

**Devuelve** A byte slice containing the marshalled JSON.

**Tipo del valor devuelto** []byte

**Devuelve** Any error that occurs.

**Tipo del valor devuelto** error

Implements the `encoding/json.Marshaler` interface.

```
func (*Calends) UnmarshalJSON(in []byte) → error
```

**Parámetros**

- **in** ([]byte) – A byte slice containing the marshalled JSON.

**Devuelve** Any error that occurs.

**Tipo del valor devuelto** error

Implements the encoding/json.Unmarshaler interface.

## 4.3 Using Calends in C/C++

Calends exposes a very small handful of things for use outside the library itself. Documented here are the parts most users will interact with.

Calends also exposes functions and types for extending the library's functionality. These are covered in the *Custom Calendars in C/C++* section.

---

**Nota:** Because C doesn't support objects, the library returns an identifier instead of an actual Calends value, keeping a reference to the internal Calends object in the Golang portions of the process space. This identifier is specific to the process in which it is generated, and is therefore only useful within that process itself. To save the value for later, use one of the marshalling functions documented below, and then the corresponding unmarshalling function to retrieve it elsewhere.

---

### 4.3.1 Create

```
long long unsigned int Calends_create_string(char *value, char *calendar, char *format)
```

```
long long unsigned int Calends_create_string_range(char *start, char *end, char *calendar, char *format)
```

```
long long unsigned int Calends_create_string_start_period(char *start, char *duration, char *calendar, char *format)
```

```
long long unsigned int Calends_create_string_end_period(char *duration, char *end, char *calendar, char *format)
```

```
long long unsigned int Calends_create_long_long(long long int value, char *calendar, char *format)
```

```
long long unsigned int Calends_create_long_long_range(long long int start, long long int end, char *calendar, char *format)
```

```
long long unsigned int Calends_create_long_long_start_period(long long int start, long long int duration, char *calendar, char *format)
```

```
long long unsigned int Calends_create_long_long_end_period(long long int duration, long long int end, char *calendar, char *format)
```

```
long long unsigned int Calends_create_double(double value, char *calendar, char *format)
```

```
long long unsigned int Calends_create_double_range(double start, double end, char *calendar, char *format)
```

```
long long unsigned int Calends_create_double_start_period(double start, double duration, char *calendar, char *format)
```

```
long long unsigned int Calends_create_double_end_period(double duration, double end, char *calendar, char *format)
```

**Parámetros**

- **value** (char\* or long long int or double) – The value to parse the date/time from.
- **start** (char\* or long long int or double) – The value to parse the start date/time from.
- **duration** (char\* or long long int or double) – The value to parse the duration from.
- **end** (char\* or long long int or double) – The value to parse the end date/time from.
- **calendar** (char\*) – The calendar system to parse the date(s)/time(s) with.
- **format** (char\*) – The format the date(s)/time(s) is/are expected to use.

**Devuelve** A new Calends object identifier

**Tipo del valor devuelto** long long unsigned int

Creates a new Calends object identifier, using *calendar* to select a calendar system, and *format* to parse the contents of value, start, *end*, and/or *duration* into the Calends object's internal instants.

### 4.3.2 Read

char \***Calends\_date**(long long unsigned int c, char \*calendar, char \*format)

**Parámetros**

- **c** (long long unsigned int) – The identifier of the current Calends object.
- **calendar** (char\*) – The calendar system to format the date/time with.
- **format** (char\*) – The format the date/time is expected to be in.

**Devuelve** The start date of the Calends object

**Tipo del valor devuelto** char\*

Retrieves the start date of the Calends object as a string. The value is generated by the calendar system given in *calendar*, according to the format string in *format*.

char \***Calends\_end\_date**(long long unsigned int c, char \*calendar, char \*format)

**Parámetros**

- **c** (long long unsigned int) – The identifier of the current Calends object.
- **calendar** (char\*) – The calendar system to format the date/time with.
- **format** (char\*) – The format the date/time is expected to be in.

**Devuelve** The end date of the Calends object

**Tipo del valor devuelto** char\*

Retrieves the end date of the Calends object as a string. The value is generated by the calendar system given in *calendar*, according to the format string in *format*.

char \***Calends\_duration**(long long unsigned int c)

**Parámetros**

- **c** (long long unsigned int) – The identifier of the current Calends object.

**Devuelve** The duration of the Calends object

**Tipo del valor devuelto** char\*

Retrieves the duration of the Calends object as a string. This value will be 0 if the Calends object is an instant.

### 4.3.3 Update

long long unsigned int **Calends\_with\_date\_string**(long long unsigned int c, char \*value, char \*calendar, char \*format)

long long unsigned int **Calends\_with\_date\_long\_long**(long long unsigned int c, long long int value, char \*calendar, char \*format)

long long unsigned int **Calends\_with\_date\_double**(long long unsigned int c, double value, char \*calendar, char \*format)

#### Parámetros

- **c** (long long unsigned int) – The identifier of the current Calends object.
- **value** (char\* or long long int or double) – The value to parse the date/time from.
- **calendar** (char\*) – The calendar system to parse the date/time with.
- **format** (char\*) – The format the date/time is expected to use.

**Devuelve** A new Calends object identifier

**Tipo del valor devuelto** long long unsigned int

Sets the start date of a Calends object, based on the Calends object's current value. The inputs are the same as for the Calends\_create\_{type}() functions, above.

long long unsigned int **Calends\_with\_end\_date\_string**(long long unsigned int c, char \*value, char \*calendar, char \*format)

long long unsigned int **Calends\_with\_end\_date\_long\_long**(long long unsigned int c, long long int value, char \*calendar, char \*format)

long long unsigned int **Calends\_with\_end\_date\_double**(long long unsigned int c, double value, char \*calendar, char \*format)

#### Parámetros

- **c** (long long unsigned int) – The identifier of the current Calends object.
- **value** (char\* or long long int or double) – The value to parse the date/time from.
- **calendar** (char\*) – The calendar system to parse the date/time with.
- **format** (char\*) – The format the date/time is expected to use.

**Devuelve** A new Calends object identifier

**Tipo del valor devuelto** long long unsigned int

Sets the end date of a Calends object, based on the Calends object's current value. The inputs are the same as for the Calends\_create\_{type}() functions, above.

long long unsigned int **Calends\_with\_duration\_string**(long long unsigned int c, char \*duration, char \*calendar)

long long unsigned int **Calends\_with\_duration\_long\_long**(long long unsigned int c, long long int duration, char \*calendar)

long long unsigned int **Calends\_with\_duration\_double**(long long unsigned int c, double duration, char \*calendar)

#### Parámetros

- **c** (long long unsigned int) – The identifier of the current Calends object.

- **duration** (char\* or long long int or double) – The value to parse the new duration from.
- **calendar** (char\*) – The calendar system to parse the date/time with.

**Devuelve** A new Calends object identifier

**Tipo del valor devuelto** long long unsigned int

Sets the duration of a Calends object, by adjusting its end point, and using the current start point as an anchor. The *duration* value is interpreted by the calendar system given in *calendar*, so is subject to any of its rules.

long long unsigned int **Calends\_with\_duration\_from\_end\_string**(long long unsigned int c, char \*duration, char \*calendar)

long long unsigned int **Calends\_with\_duration\_from\_end\_long\_long**(long long unsigned int c, long long int duration, char \*calendar)

long long unsigned int **Calends\_with\_duration\_from\_end\_double**(long long unsigned int c, double duration, char \*calendar)

#### Parámetros

- **c** (long long unsigned int) – The identifier of the current Calends object.
- **duration** (char\* or long long int or double) – The value to parse the new duration from.
- **calendar** (char\*) – The calendar system to parse the date/time with.

**Devuelve** A new Calends object identifier

**Tipo del valor devuelto** long long unsigned int

Sets the duration of a Calends object, by adjusting its start point, and using the current end point as an anchor. The *duration* value is interpreted by the calendar system given in *calendar*, so is subject to any of its rules.

### 4.3.4 Destroy

void **Calends\_release**(long long unsigned int c)

#### Parámetros

- **c** (long long unsigned int) – The identifier of the current Calends object.

Releases an internal Calends object, freeing its associated memory. Since the memory used in this case is kept within the Golang portion of the process space, we don't have access to free that memory using more conventional C/C++ methods, so this function offers that functionality instead.

### 4.3.5 Manipulate

long long unsigned int **Calends\_add\_string**(long long unsigned int c, char \*offset, char \*calendar)

long long unsigned int **Calends\_add\_long\_long**(long long unsigned int c, long long int offset, char \*calendar)

long long unsigned int **Calends\_add\_double**(long long unsigned int c, double offset, char \*calendar)

#### Parámetros

- **c** (long long unsigned int) – The identifier of the current Calends object.
- **offset** (char\*) – The value to parse the offset from.
- **calendar** (char\*) – The calendar system to parse the date/time with.

**Devuelve** A new Calends object identifier

**Tipo del valor devuelto** long long unsigned int

Increases the end date of the Calends object's current value by *offset*, as interpreted by the calendar system given in *calendar*.

long long unsigned int **Calends\_add\_from\_end\_string**(long long unsigned int c, char \*offset, char \*calendar)

long long unsigned int **Calends\_add\_from\_end\_long\_long**(long long unsigned int c, long long int offset, char \*calendar)

long long unsigned int **Calends\_add\_from\_end\_double**(long long unsigned int c, double offset, char \*calendar)

**Parámetros**

- **c** (long long unsigned int) – The identifier of the current Calends object.
- **offset** (char\*) – The value to parse the offset from.
- **calendar** (char\*) – The calendar system to parse the date/time with.

**Devuelve** A new Calends object identifier

**Tipo del valor devuelto** long long unsigned int

Increases the start date of the Calends object's current value by *offset*, as interpreted by the calendar system given in *calendar*.

long long unsigned int **Calends\_subtract\_string**(long long unsigned int c, char \*offset, char \*calendar)

long long unsigned int **Calends\_subtract\_long\_long**(long long unsigned int c, long long int offset, char \*calendar)

long long unsigned int **Calends\_subtract\_double**(long long unsigned int c, double offset, char \*calendar)

**Parámetros**

- **c** (long long unsigned int) – The identifier of the current Calends object.
- **offset** (char\*) – The value to parse the offset from.
- **calendar** (char\*) – The calendar system to parse the date/time with.

**Devuelve** A new Calends object identifier

**Tipo del valor devuelto** long long unsigned int

Works the same as **Calends\_add\_{type}()**, except it decreases the start date, rather than increasing it.

long long unsigned int **Calends\_subtract\_from\_end\_string**(long long unsigned int c, char \*offset, char \*calendar)

long long unsigned int **Calends\_subtract\_from\_end\_long\_long**(long long unsigned int c, long long int offset, char \*calendar)

long long unsigned int **Calends\_subtract\_from\_end\_double**(long long unsigned int c, double offset, char \*calendar)

**Parámetros**

- **c** (long long unsigned int) – The identifier of the current Calends object.
- **offset** (char\*) – The value to parse the offset from.
- **calendar** (char\*) – The calendar system to parse the date/time with.

**Devuelve** A new Calends object identifier

**Tipo del valor devuelto** long long unsigned int

Works the same as `Calends_add_from_end_{type}()`, except it decreases the end date, rather than increasing it.

`long long unsigned int Calends_next_string(long long unsigned int c, char *offset, char *calendar)`

`long long unsigned int Calends_next_long_long(long long unsigned int c, long long int offset, char *calendar)`

`long long unsigned int Calends_next_double(long long unsigned int c, double offset, char *calendar)`

#### Parámetros

- **c** (`long long unsigned int`) – The identifier of the current Calends object.
- **offset** (`char*`) – The value to parse the offset from.
- **calendar** (`char*`) – The calendar system to parse the date/time with.

**Devuelve** A new Calends object identifier

**Tipo del valor devuelto** `long long unsigned int`

Returns the identifier of a Calends object of *offset* duration (as interpreted by the calendar system given in *calendar*), which abuts the current Calends object's start value. If *offset* is empty, *calendar* is ignored, and the current object's duration is used instead.

`long long unsigned int Calends_previous_string(long long unsigned int c, char *offset, char *calendar)`

`long long unsigned int Calends_previous_long_long(long long unsigned int c, long long int offset, char *calendar)`

`long long unsigned int Calends_previous_double(long long unsigned int c, double offset, char *calendar)`

#### Parámetros

- **c** (`long long unsigned int`) – The identifier of the current Calends object.
- **offset** (`char*`) – The value to parse the offset from.
- **calendar** (`char*`) – The calendar system to parse the date/time with.

**Devuelve** A new Calends object identifier

**Tipo del valor devuelto** `long long unsigned int`

Returns the identifier of a Calends object of *offset* duration (as interpreted by the calendar system given in *calendar*), which abuts the current Calends object's end value. If *offset* is empty, *calendar* is ignored, and the current object's duration is used instead.

### 4.3.6 Combine

`long long unsigned int Calends_merge(long long unsigned int c1, long long unsigned int c2)`

#### Parámetros

- **c1** (`long long unsigned int`) – The identifier of the current Calends object.
- **c2** (`long long unsigned int`) – The identifier of the Calends object to merge.

**Devuelve** A new Calends object identifier

**Tipo del valor devuelto** `long long unsigned int`

Returns a Calends object spanning from the earliest start date to the latest end date between *c1* and *c2*.

long long unsigned int **Calends\_intersect**(long long unsigned int c1, long long unsigned int c2)

**Parámetros**

- **c1** (long long unsigned int) – The identifier of the current Calends object.
- **c2** (long long unsigned int) – The identifier of the Calends object to intersect.

**Devuelve** A new Calends object identifier

**Tipo del valor devuelto** long long unsigned int

Returns a Calends object spanning the overlap between **c1** and **c2**. If **c1** and **c2** don't overlap, returns an error.

long long unsigned int **Calends\_gap**(long long unsigned int c1, long long unsigned int c2)

**Parámetros**

- **c1** (long long unsigned int) – The identifier of the current Calends object.
- **c2** (long long unsigned int) – The identifier of the Calends object to gap.

**Devuelve** A new Calends object identifier

**Tipo del valor devuelto** long long unsigned int

Returns a Calends object spanning the gap between **c1** and **c2**. If **c1** and **c2** overlap (and there is, therefore, no gap to return), returns an error.

## 4.3.7 Compare

char \***Calends\_difference**(long long unsigned int c1, long long unsigned int c2, char \*mode)

**Parámetros**

- **c1** (long long unsigned int) – The identifier of the current Calends object.
- **c2** (long long unsigned int) – The identifier of the Calends object to compare.
- **mode** (char\*) – The comparison mode.

**Devuelve** The difference, as a decimal string

**Tipo del valor devuelto** char\*

Returns the difference of **c1** minus **c2**, using **mode** to select which values to use in the calculation. Valid **modes** include:

- "start" - use the start date of both **c1** and **c2**
- "duration" - use the duration of both **c1** and **c2**
- "end" - use the end date of both **c1** and **c2**
- "start-end" - use the start of **c1**, and the end of **c2**
- "end-start" - use the end of **c1**, and the start of **c2**

signed char **Calends\_compare**(long long unsigned int c1, long long unsigned int c2, char \*mode)

**Parámetros**

- **c1** (long long unsigned int) – The identifier of the current Calends object.
- **c2** (long long unsigned int) – The identifier of the Calends object to compare.
- **mode** (char\*) – The comparison mode.

**Devuelve** A standard comparison result

**Tipo del valor devuelto** signed char

Returns -1 if *c1* is less than *c2*, 0 if they are equal, and 1 if *c1* is greater than *c2*, using *mode* to select which values to use in the comparison. Valid *modes* are the same as for *Calends\_difference()*, above.

signed char **Calends\_contains**(long long unsigned int c1, long long unsigned int c2)

**Parámetros**

- *c1* (long long unsigned int) – The identifier of the current Calends object.
- *c2* (long long unsigned int) – The identifier of the Calends object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** signed char

Checks whether *c1* contains all of *c2*.

signed char **Calends\_overlaps**(long long unsigned int c1, long long unsigned int c2)

**Parámetros**

- *c1* (long long unsigned int) – The identifier of the current Calends object.
- *c2* (long long unsigned int) – The identifier of the Calends object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** signed char

Checks whether *c1* overlaps with *c2*.

signed char **Calends\_abuts**(long long unsigned int c1, long long unsigned int c2)

**Parámetros**

- *c1* (long long unsigned int) – The identifier of the current Calends object.
- *c2* (long long unsigned int) – The identifier of the Calends object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** signed char

Checks whether *c1* abuts *c2* (that is, whether one begins at the same instant the other ends).

signed char **Calends\_is\_same**(long long unsigned int c1, long long unsigned int c2)

**Parámetros**

- *c1* (long long unsigned int) – The identifier of the current Calends object.
- *c2* (long long unsigned int) – The identifier of the Calends object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** signed char

Checks whether *c1* covers the same span of time as *c2*.

signed char **Calends\_is\_shorter**(long long unsigned int c1, long long unsigned int c2)

**Parámetros**

- *c1* (long long unsigned int) – The identifier of the current Calends object.
- *c2* (long long unsigned int) – The identifier of the Calends object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** signed char

Compares the duration of *c1* and *c2*.

signed char **Calends\_is\_same\_duration**(long long unsigned int c1, long long unsigned int c2)

**Parámetros**

- **c1** (long long unsigned int) – The identifier of the current Calends object.
- **c2** (long long unsigned int) – The identifier of the Calends object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** signed char

Compares the duration of *c1* and *c2*.

signed char **Calends\_is\_longer**(long long unsigned int c1, long long unsigned int c2)

**Parámetros**

- **c1** (long long unsigned int) – The identifier of the current Calends object.
- **c2** (long long unsigned int) – The identifier of the Calends object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** signed char

Compares the duration of *c1* and *c2*.

signed char **Calends\_is\_before**(long long unsigned int c1, long long unsigned int c2)

**Parámetros**

- **c1** (long long unsigned int) – The identifier of the current Calends object.
- **c2** (long long unsigned int) – The identifier of the Calends object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** signed char

Compares the entirety of *c1* with the start date of *c2*.

signed char **Calends\_starts\_before**(long long unsigned int c1, long long unsigned int c2)

**Parámetros**

- **c1** (long long unsigned int) – The identifier of the current Calends object.
- **c2** (long long unsigned int) – The identifier of the Calends object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** signed char

Compares the start date of *c1* with the start date of *c2*.

signed char **Calends\_ends\_before**(long long unsigned int c1, long long unsigned int c2)

**Parámetros**

- **c1** (long long unsigned int) – The identifier of the current Calends object.
- **c2** (long long unsigned int) – The identifier of the Calends object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** signed char

Compares the end date of *c1* with the start date of *c2*.

signed char **Calends\_is\_during**(long long unsigned int c1, long long unsigned int c2)

**Parámetros**

- **c1** (long long unsigned int) – The identifier of the current Calends object.
- **c2** (long long unsigned int) – The identifier of the Calends object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** signed char

Checks whether the entirety of *c1* lies between the start and end dates of *c2*.

signed char **Calends\_starts\_during**(long long unsigned int c1, long long unsigned int c2)

**Parámetros**

- **c1** (long long unsigned int) – The identifier of the current Calends object.
- **c2** (long long unsigned int) – The identifier of the Calends object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** signed char

Checks whether the start date of *c1* lies between the start and end dates of *c2*.

signed char **Calends\_ends\_during**(long long unsigned int c1, long long unsigned int c2)

**Parámetros**

- **c1** (long long unsigned int) – The identifier of the current Calends object.
- **c2** (long long unsigned int) – The identifier of the Calends object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** signed char

Checks whether the end date of *c1* lies between the start and end dates of *c2*.

signed char **Calends\_is\_after**(long long unsigned int c1, long long unsigned int c2)

**Parámetros**

- **c1** (long long unsigned int) – The identifier of the current Calends object.
- **c2** (long long unsigned int) – The identifier of the Calends object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** signed char

Compares the entirety of *c1* with the end date of *c2*.

signed char **Calends\_starts\_after**(long long unsigned int c1, long long unsigned int c2)

**Parámetros**

- **c1** (long long unsigned int) – The identifier of the current Calends object.
- **c2** (long long unsigned int) – The identifier of the Calends object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** signed char

Compares the start date of [c1](#) with the end date of [c2](#).

signed char **Calends\_ends\_after**(long long unsigned int c1, long long unsigned int c2)

**Parámetros**

- **c1** (long long unsigned int) – The identifier of the current Calends object.
- **c2** (long long unsigned int) – The identifier of the Calends object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** signed char

Compares the end date of [c1](#) with the end date of [c2](#).

### 4.3.8 Export

char \***Calends\_string**(long long unsigned int c)

**Parámetros**

- **c** (long long unsigned int) – The Calends object to convert.

**Devuelve** The string representation of the current value.

**Tipo del valor devuelto** char\*

Converts the value of [c](#) to a string.

char \***Calends\_encode\_text**(long long unsigned int c)

**Parámetros**

- **c** (long long unsigned int) – The Calends object to encode.

**Devuelve** The encoded representation of the current value.

**Tipo del valor devuelto** char\*

Encodes the value of [c](#) as text, for external storage.

long long unsigned int **Calends\_decode\_text**(char \*in)

**Parámetros**

- **in** (char\*) – The encoded representation of a Calends value.

**Devuelve** The decoded Calends object's identifier.

**Tipo del valor devuelto** long long unsigned int

Decodes the value of [in](#) to a new Calends object.

char \***Calends\_encode\_json**(long long unsigned int c)

**Parámetros**

- **c** (long long unsigned int) – The Calends object to encode.

**Devuelve** The encoded representation of the current value.

**Tipo del valor devuelto** char\*

Encodes the value of `c` as JSON, for external communication.

long long unsigned int **Calends\_decode\_json**(char \*in)

**Parámetros**

- **in** (char\*) – The encoded representation of a Calends value.

**Devuelve** The decoded Calends object's identifier.

**Tipo del valor devuelto** long long unsigned int

Decodes the value of `in` to a new Calends object.

### 4.3.9 Error Handling

void **Calends\_register\_panic\_handler**(Calends\_panic\_handler callback)

**Parámetros**

- **callback** (void function(char\*)) – A panic handler function.

When errors happen, Go normally returns the error as an additional return value. Since most programming languages don't support this, the C/C++ interface to the library instead relies on a Golang feature called a **panic**, which is a lot like an exception in many other languages. This function allows users to register a callback function of their own to handle the error conditions which might come up. `callback` should accept a `char*` containing the error message supplied by the library, and return nothing.

## 4.4 Using Calends in Dart

Calends exposes a very small handful of things for use outside the library itself. One is the `Calends` class, documented here, which should be the only interface users of the library ever need to touch.

Another thing Calends exposes is the `TAI64Time` class, used to store and manipulate the instants of time which make up a `Calends` instance. The rest are interfaces for extending the library's functionality. These are covered in the *Custom Calendars in Dart* section.

---

**Nota:** `Calends` objects are immutable - all methods return a new `Calends` object where they might otherwise alter the current one. This has the side effect of using more memory to perform manipulations than updating values on an existing object would. It makes many operations safer, though, than mutable objects would allow.

---

**class** `calends.Calends`

The main entry point to Calends and its functionality.

`Calends.initialize()`

Sets up the error handling and otherwise preps the library for use. Run this before doing anything else!

### 4.4.1 Create

`Calends.(stamp, calendar, format)`

#### Argumentos

- **stamp** (`dynamic()`) – The timestamp to parse the date(s)/time(s) from.
- **calendar** (`String()`) – The calendar system to parse the date(s)/time(s) with.
- **format** (`String()`) – The format the date(s)/time(s) is/are expected to use.

**Lanzamientos** `CalendsException()` – when an error occurs

Creates a new `Calends` object, using `calendar` to select a calendar system, and `format` to parse the contents of `stamp` into the `Calends` object's internal instants. The type of `stamp` can vary based on the calendar system itself, but generally speaking can always be a string.

In any case, the `stamp` can also be a `Map`, where the `String` keys are any two of `start`, `end`, and `duration`. If all three are provided, `duration` is ignored in favor of calculating it directly.

The calendar system then converts `stamp` to a pair of `TAI64Time` instants, which the `Calends` object sets to the appropriate internal value.

### 4.4.2 Read

`Calends.date(calendar, format)`

#### Argumentos

- **calendar** (`String()`) – The calendar system to format the date/time with.
- **format** (`String()`) – The format the date/time is expected to be in.

**Devuelve** The start date of the `Calends` object

**Tipo del valor devuelto** `string`

**Lanzamientos** `CalendsException()` – when an error occurs

Retrieves the start date of the `Calends` object as a string. The value is generated by the calendar system given in `calendar`, according to the format string in `format`.

`Calends.endDate(calendar, format)`

#### Argumentos

- **calendar** (`String()`) – The calendar system to format the date/time with.
- **format** (`String()`) – The format the date/time is expected to be in.

**Devuelve** The end date of the `Calends` object

**Tipo del valor devuelto** `string`

**Lanzamientos** `CalendsException()` – when an error occurs

Retrieves the end date of the `Calends` object as a string. The value is generated by the calendar system given in `calendar`, according to the format string in `format`.

`Calends.duration()`

**Devuelve** The duration of the `Calends` object

**Tipo del valor devuelto** `string`

Retrieves the duration of the *Calends* object as a decimal string. This value will be 0 if the *Calends* object contains an instant.

### 4.4.3 Update

*Calends.withDate*(stamp, calendar, format)

#### Argumentos

- **stamp** (dynamic()) – The value to parse the date/time from.
- **calendar** (String()) – The calendar system to parse the date/time with.
- **format** (String()) – The format the date/time is expected to use.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Lanzamientos** *CalendsException()* – when an error occurs

Returns a *Calends* object with a start date based on the current *Calends* object's value. The inputs are the same as for *Calends.()*, above, except the *String* → value map option isn't available, since you're already specifically setting the start value explicitly.

*Calends.withEndDate*(value, calendar, format)

#### Argumentos

- **stamp** (dynamic()) – The value to parse the date/time from.
- **calendar** (String()) – The calendar system to parse the date/time with.
- **format** (String()) – The format the date/time is expected to use.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Lanzamientos** *CalendsException()* – when an error occurs

Returns a *Calends* object with an end date based on the current *Calends* object's value. The inputs are the same as for *Calends.()*, above, except the *String* → value map option isn't available, since you're already specifically setting the end value explicitly.

*Calends.withDuration*(duration, calendar)

#### Argumentos

- **duration** (dynamic()) – The value to parse the new duration from.
- **calendar** (String()) – The calendar system to parse the date/time with.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Lanzamientos** *CalendsException()* – when an error occurs

Returns a *Calends* object with a duration set by adjusting the current *Calends* object's end point, and using its start point as an anchor. The duration value is interpreted by the calendar system given in calendar, so is subject to any of its rules.

`Calends.withDurationFromEnd(duration, calendar)`

**Argumentos**

- **duration** (dynamic()) – The value to parse the new duration from.
- **calendar** (String()) – The calendar system to parse the date/time with.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Lanzamientos** *CalendsException()* – when an error occurs

Returns a *Calends* object with a duration set by adjusting the current *Calends* object's start point, and using its end point as an anchor. The *duration* value is interpreted by the calendar system given in *calendar*, so is subject to any of its rules.

## 4.4.4 Manipulate

`Calends.add(offset, calendar)`

**Argumentos**

- **offset** (dynamic()) – The value to parse the offset from.
- **calendar** (String()) – The calendar system to parse the date/time with.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Lanzamientos** *CalendsException()* – when an error occurs

Increases the end date of the *Calends* object's current value by *offset*, as interpreted by the calendar system given in *calendar*, and returns a new *Calends* object with the result.

`Calends.addFromEnd(offset, calendar)`

**Argumentos**

- **offset** (dynamic()) – The value to parse the offset from.
- **calendar** (String()) – The calendar system to parse the date/time with.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Lanzamientos** *CalendsException()* – when an error occurs

Increases the start date of the *Calends* object's current value by *offset*, as interpreted by the calendar system given in *calendar*, and returns a new *Calends* object with the result.

`Calends.subtract(offset, calendar)`

**Argumentos**

- **offset** (dynamic()) – The value to parse the offset from.
- **calendar** (String()) – The calendar system to parse the date/time with.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Lanzamientos** `CalendsException()` – when an error occurs

Works the same as `add()`, except it decreases the start date, rather than increasing it.

`Calends.subtractFromEnd(offset, calendar)`

**Argumentos**

- **offset** (`dynamic()`) – The value to parse the offset from.
- **calendar** (`String()`) – The calendar system to parse the date/time with.

**Devuelve** A new `Calends` object

**Tipo del valor devuelto** `Calends`

**Lanzamientos** `CalendsException()` – when an error occurs

Works the same as `addFromEnd()`, except it decreases the end date, rather than increasing it.

`Calends.next(offset, calendar)`

**Argumentos**

- **offset** (`dynamic()`) – The value to parse the offset from.
- **calendar** (`String()`) – The calendar system to parse the date/time with.

**Devuelve** A new `Calends` object

**Tipo del valor devuelto** `Calends`

**Lanzamientos** `CalendsException()` – when an error occurs

Returns a `Calends` object of `offset` duration (as interpreted by the calendar system given in `calendar`), which abuts the current `Calends` object's value. If `offset` is empty, `calendar` is ignored, and the current object's duration is used instead.

`Calends.previous(offset, calendar)`

**Argumentos**

- **offset** (`dynamic()`) – The value to parse the offset from.
- **calendar** (`String()`) – The calendar system to parse the date/time with.

**Devuelve** A new `Calends` object

**Tipo del valor devuelto** `Calends`

**Lanzamientos** `CalendsException()` – when an error occurs

Returns a `Calends` object of `offset` duration (as interpreted by the calendar system given in `calendar`), which abuts the current `Calends` object's value. If `offset` is empty, `calendar` is ignored, and the current object's duration is used instead.

### 4.4.5 Combine

`Calends.merge(c2)`

**Argumentos**

- `c2` (*Calends()*) – The *Calends* object to merge.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Lanzamientos** *CalendsException()* – when an error occurs

Returns a *Calends* object spanning from the earliest start date to the latest end date between the current *Calends* object and `c2`.

`Calends.intersect(c2)`

**Argumentos**

- `c2` (*Calends()*) – The *Calends* object to intersect.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Lanzamientos** *CalendsException()* – when an error occurs

Returns a *Calends* object spanning the overlap between the current *Calends* object and `c2`. If the current object and `c2` don't overlap, throws an error.

`Calends.gap(c2)`

**Argumentos**

- `c2` (*Calends()*) – The *Calends* object to gap.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Lanzamientos** *CalendsException()* – when an error occurs

Returns a *Calends* object spanning the gap between the current *Calends* object and `c2`. If the current object and `c2` overlap (and there is, therefore, no gap to return), throws an error.

### 4.4.6 Compare

`Calends.difference(c2, mode)`

**Argumentos**

- `c2` (*Calends()*) – The *Calends* object to compare.
- `mode` (*String()*) – The comparison mode.

**Devuelve** The difference, as a decimal string

**Tipo del valor devuelto** *string*

Returns the difference of the current *Calends* object minus `c2`, using `mode` to select which values to use in the calculation. Valid modes include:

- `start` - use the start date of both the current object and `c2`

- **duration** - use the duration of both the current object and `c2`
- **end** - use the end date of both the current object and `c2`
- **start-end** - use the start of the current object, and the end of `c2`
- **end-start** - use the end of the current object, and the start of `c2`

`Calends.compare(c2, mode)`

**Argumentos**

- **c2** (`Calends()`) – The `Calends` object to compare.
- **mode** (`String()`) – The comparison mode.

**Devuelve** A standard comparison result

**Tipo del valor devuelto** `int`

Returns `-1` if the current `Calends` object is less than `c2`, `0` if they are equal, and `1` if the current object is greater than `c2`, using `mode` to select which values to use in the comparison. Valid modes are the same as for `difference()`, above.

`Calends.contains(c2)`

**Argumentos**

- **c2** (`Calends()`) – The `Calends` object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** `bool`

Checks whether the current `Calends` object contains all of `c2`.

`Calends.overlaps(c2)`

**Argumentos**

- **c2** (`Calends()`) – The `Calends` object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** `bool`

Checks whether the current `Calends` object overlaps with `c2`.

`Calends.abuts(c2)`

**Argumentos**

- **c2** (`Calends()`) – The `Calends` object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** `bool`

Checks whether the current `Calends` object abuts `c2` (that is, whether one begins at the same instant the other ends).

`Calends.isSame(c2)`

**Argumentos**

- **c2** (`Calends()`) – The `Calends` object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Checks whether the current *Calends* object covers the same span of time as c2. Also available via the == operator.

`Calends.isShorter(c2)`

**Argumentos**

- `c2 (Calends())` – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the duration of the current *Calends* object and c2.

`Calends.isSameDuration(c2)`

**Argumentos**

- `c2 (Calends())` – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the duration of the current *Calends* object and c2.

`Calends.isLonger(c2)`

**Argumentos**

- `c2 (Calends())` – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the duration of the current *Calends* object and c2.

`Calends.isBefore(c2)`

**Argumentos**

- `c2 (Calends())` – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the entirety of the current *Calends* object with the start date of c2. Also available as the < operator.

`Calends.startsBefore(c2)`

**Argumentos**

- `c2 (Calends())` – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the start date of the current *Calends* object with the start date of c2.

`Calends.endsBefore(c2)`

**Argumentos**

- `c2` (*Calends()*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the end date of the current *Calends* object with the start date of `c2`.

`Calends.isDuring(c2)`

**Argumentos**

- `c2` (*Calends()*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Checks whether the entirety of the current *Calends* object lies between the start and end dates of `c2`.

`Calends.startsDuring(c2)`

**Argumentos**

- `c2` (*Calends()*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Checks whether the start date of the current *Calends* object lies between the start and end dates of `c2`.

`Calends.endsDuring(c2)`

**Argumentos**

- `c2` (*Calends()*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Checks whether the end date of the current *Calends* object lies between the start and end dates of `c2`.

`Calends.isAfter(c2)`

**Argumentos**

- `c2` (*Calends()*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the entirety of the current *Calends* object with the end date of `c2`. Also available as the `>` operator.

`Calends.startsAfter(c2)`

**Argumentos**

- `c2` (*Calends()*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** `bool`

Compares the start date of the current *Calends* object with the end date of `c2`.

`Calends.endsAfter(c2)`

**Argumentos**

- `c2` (*Calends()*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** `bool`

Compares the end date of the current *Calends* object with the end date of `c2`.

## 4.4.7 Export

It's possible to export *Calends* values in a couple of ways, and then re-import them later/elsewhere. The text encoding is the less portable option, though, so we strongly recommend using JSON instead.

`Calends.encodeText()`

**Devuelve** The text-encoded value of the *Calends* instance.

**Tipo del valor devuelto** `String`

**Lanzamientos** *CalendsException()* – when an error occurs

`Calends.decodeText(encoded)`

**Argumentos**

- `encoded` (`String()`) – The text-encoded value to import.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Lanzamientos** *CalendsException()* – when an error occurs

`Calends.encodeJson()`

**Devuelve** The JSON-encoded value of the *Calends* instance.

**Tipo del valor devuelto** `String`

**Lanzamientos** *CalendsException()* – when an error occurs

`Calends.decodeJson(encoded)`

**Argumentos**

- `encoded` (`String()`) – The JSON-encoded value to import.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Lanzamientos** `CalendsException()` – when an error occurs

If you just need a `String` value for display purposes, and can't use the `date()` method to format it to a given calendar system, there is a fallback mechanism in the form of the `toString()` method. This value is neither human-readable nor machine-importable, so it should only be used for debugging.

`Calends.toString()`

**Devuelve** A non-portable representation of the `Calends` instance.

**Tipo del valor devuelto** `String`

**Lanzamientos** `CalendsException()` – when an error occurs

## 4.4.8 Error Handling

**class** `calends.CalendsException`

A very simple exception class, directly extending `Exception`. It is thrown by the library for all encountered errors.

## 4.5 Using Calends in JS/WASM

Calends exposes a very small handful of things for use outside the library itself. One is the `Calends()` class, documented here, which should be the only interface users of the library ever need to touch.

Another thing Calends exposes is the `TAI64Time()` class, used to store and manipulate the instants of time which make up a `Calends()` moment or instance. The rest are interfaces for extending the library's functionality. These are covered in the *Custom Calendars in JS/WASM* section.

---

**Nota:** `Calends()` objects are immutable - all methods return a new `Calends()` object where they might otherwise alter the current one. This has the side effect of using more memory to perform manipulations than updating values on an existing object would. It makes many operations safer, though, than mutable objects would allow.

---

### 4.5.1 Create

**class** `calends.Calends(stamp, calendar, format)`

**Argumentos**

- **stamp** (`mixed()`) – The value to parse the date(s)/time(s) from.
- **calendar** (`string()`) – The calendar system to parse the date(s)/time(s) with.
- **format** (`string()`) – The format the date(s)/time(s) is/are expected to use.

**Devuelve** A new `Calends()` object

**Tipo del valor devuelto** `Calends()`

**Lanzamientos** `CalendsException()` – when an error occurs

Creates a new `Calends()` object, using `calendar` to select a calendar system, and `format` to parse the contents of `stamp` into the `Calends()` object's internal instants. The type of `stamp` can vary based on the calendar system itself, but generally speaking can always be a string.

In any case, the value can always be an associative array, where the keys are any two of `start`, `end`, and `duration`. If all three are provided, `duration` is ignored in favor of calculating it directly.

The calendar system then converts `stamp` to a `TAI64Time()` instant, which the `Calends()` object sets to the appropriate internal value.

### 4.5.2 Read

`Calends.date(calendar, format)`

**Argumentos**

- **calendar** (string()) – The calendar system to format the date/time with.
- **format** (string()) – The format the date/time is expected to be in.

**Devuelve** The start date of the `Calends()` object

**Tipo del valor devuelto** string

**Lanzamientos** `CalendsException()` – when an error occurs

Retrieves the start date of the `Calends()` object as a string. The value is generated by the calendar system given in `calendar`, according to the format string in `format`.

`Calends.endDate(calendar, format)`

**Argumentos**

- **calendar** (string()) – The calendar system to format the date/time with.
- **format** (string()) – The format the date/time is expected to be in.

**Devuelve** The end date of the `Calends()` object

**Tipo del valor devuelto** string

**Lanzamientos** `CalendsException()` – when an error occurs

Retrieves the end date of the `Calends()` object as a string. The value is generated by the calendar system given in `calendar`, according to the format string in `format`.

`Calends.duration()`

**Devuelve** The duration of the `Calends()` object

**Tipo del valor devuelto** string

Retrieves the duration of the `Calends()` object as a decimal string. This value will be `0` if the `Calends()` object is an instant.

### 4.5.3 Update

`Calends.withDate(stamp, calendar, format)`

**Argumentos**

- **stamp** (mixed()) – The value to parse the date/time from.
- **calendar** (string()) – The calendar system to parse the date/time with.
- **format** (string()) – The format the date/time is expected to use.

**Devuelve** A new `Calends()` object

**Tipo del valor devuelto** `Calends()`

**Lanzamientos** `CalendsException()` – when an error occurs

Returns a `Calends()` object with a start date based on the current `Calends()` object's value. The inputs are the same as for `Calends()`, above, except the string  $\rightarrow$  value map option isn't available, since you're already specifically setting the start value explicitly.

`Calends.withEndDate(stamp, calendar, format)`

**Argumentos**

- **stamp** (`mixed()`) – The value to parse the date/time from.
- **calendar** (`string()`) – The calendar system to parse the date/time with.
- **format** (`string()`) – The format the date/time is expected to use.

**Devuelve** A new `Calends()` object

**Tipo del valor devuelto** `Calends()`

**Lanzamientos** `CalendsException()` – when an error occurs

Returns a `Calends()` object with an end date based on the current `Calends()` object's value. The inputs are the same as for `Calends()`, above, except the string  $\rightarrow$  value map option isn't available, since you're already specifically setting the end value explicitly.

`Calends.withDuration(duration, calendar)`

**Argumentos**

- **duration** (`string()`) – The value to parse the new duration from.
- **calendar** (`string()`) – The calendar system to parse the date/time with.

**Devuelve** A new `Calends()` object

**Tipo del valor devuelto** `Calends()`

**Lanzamientos** `CalendsException()` – when an error occurs

Returns a `Calends()` object with a duration set by adjusting the current `Calends()` object's end point, and using its start point as an anchor. The duration value is interpreted by the calendar system given in `calendar`, so is subject to any of its rules.

`Calends.withDurationFromEnd(duration, calendar)`

**Argumentos**

- **duration** (`string()`) – The value to parse the new duration from.
- **calendar** (`string()`) – The calendar system to parse the date/time with.

**Devuelve** A new `Calends()` object

**Tipo del valor devuelto** `Calends()`

**Lanzamientos** `CalendsException()` – when an error occurs

Returns a `Calends()` object with a duration set by adjusting the current `Calends()` object's start point, and using its end point as an anchor. The duration value is interpreted by the calendar system given in `calendar`, so is subject to any of its rules.

## 4.5.4 Manipulate

`Calends.add(offset, calendar)`

### Argumentos

- **offset** (string()) – The value to parse the offset from.
- **calendar** (string()) – The calendar system to parse the date/time with.

**Devuelve** A new `Calends()` object

**Tipo del valor devuelto** `Calends()`

**Lanzamientos** `CalendsException()` – when an error occurs

Increases the end date of the `Calends()` object's current value by `offset`, as interpreted by the calendar system given in `calendar`, and returns a new `Calends()` object with the result.

`Calends.addFromEnd(offset, calendar)`

### Argumentos

- **offset** (string()) – The value to parse the offset from.
- **calendar** (string()) – The calendar system to parse the date/time with.

**Devuelve** A new `Calends()` object

**Tipo del valor devuelto** `Calends()`

**Lanzamientos** `CalendsException()` – when an error occurs

Increases the start date of the `Calends()` object's current value by `offset`, as interpreted by the calendar system given in `calendar`, and returns a new `Calends()` object with the result.

`Calends.subtract(offset, calendar)`

### Argumentos

- **offset** (string()) – The value to parse the offset from.
- **calendar** (string()) – The calendar system to parse the date/time with.

**Devuelve** A new `Calends()` object

**Tipo del valor devuelto** `Calends()`

**Lanzamientos** `CalendsException()` – when an error occurs

Works the same as `add()`, except it decreases the start date, rather than increasing the end date.

`Calends.subtractFromEnd(offset, calendar)`

### Argumentos

- **offset** (string()) – The value to parse the offset from.
- **calendar** (string()) – The calendar system to parse the date/time with.

**Devuelve** A new `Calends()` object

**Tipo del valor devuelto** `Calends()`

**Lanzamientos** `CalendsException()` – when an error occurs

Works the same as `addFromEnd()`, except it decreases the end date, rather than increasing the start date.

`Calends.next(offset, calendar)`

**Argumentos**

- **offset** (string()) – The value to parse the offset from.
- **calendar** (string()) – The calendar system to parse the date/time with.

**Devuelve** A new `Calends()` object

**Tipo del valor devuelto** `Calends()`

**Lanzamientos** `CalendsException()` – when an error occurs

Returns a `Calends()` object of `offset` duration (as interpreted by the calendar system given in `calendar`), which abuts the current `Calends()` object's value. If `offset` is empty, `calendar` is ignored, and the current object's duration is used instead.

`Calends.previous(offset, calendar)`

**Argumentos**

- **offset** (string()) – The value to parse the offset from.
- **calendar** (string()) – The calendar system to parse the date/time with.

**Devuelve** A new `Calends()` object

**Tipo del valor devuelto** `Calends()`

**Lanzamientos** `CalendsException()` – when an error occurs

Returns a `Calends()` object of `offset` duration (as interpreted by the calendar system given in `calendar`), which abuts the current `Calends()` object's value. If `offset` is empty, `calendar` is ignored, and the current object's duration is used instead.

## 4.5.5 Combine

`Calends.merge(other)`

**Argumentos**

- **other** (`Calends()`) – The `Calends()` object to merge.

**Devuelve** A new `Calends()` object

**Tipo del valor devuelto** `Calends()`

**Lanzamientos** `CalendsException()` – when an error occurs

Returns a `Calends()` object spanning from the earliest start date to the latest end date between the current `Calends()` object and `other`.

`Calends.intersect(other)`

**Argumentos**

- **other** (`Calends()`) – The `Calends()` object to intersect.

**Devuelve** A new `Calends()` object

**Tipo del valor devuelto** `Calends()`

**Lanzamientos** `CalendsException()` – when an error occurs

Returns a *Calends()* object spanning the overlap between the current *Calends()* object and *other*. If the current object and *other* don't overlap, throws an error.

**Calends.gap(*other*)**

**Argumentos**

- **other** (*Calends()*) – The *Calends()* object to gap.

**Devuelve** A new *Calends()* object

**Tipo del valor devuelto** *Calends()*

**Lanzamientos** *CalendsException()* – when an error occurs

Returns a *Calends()* object spanning the gap between the current *Calends()* object and *other*. If the current object and *other* overlap (and there is, therefore, no gap to return), throws an error.

## 4.5.6 Compare

**Calends.difference(*other*, *mode*)**

**Argumentos**

- **other** (*Calends()*) – The *Calends()* object to compare.
- **mode** (*string()*) – The comparison mode.

**Devuelve** The difference, as a decimal string

**Tipo del valor devuelto** *string*

Returns the difference of the current *Calends()* object minus *other*, using *mode* to select which values to use in the calculation. Valid modes include:

- **start** - use the start date of both the current object and *other*
- **duration** - use the duration of both the current object and *other*
- **end** - use the end date of both the current object and *other*
- **start-end** - use the start of the current object, and the end of *other*
- **end-start** - use the end of the current object, and the start of *other*

**Calends.compare(*other*, *mode*)**

**Argumentos**

- **other** (*Calends()*) – The *Calends()* object to compare.
- **mode** (*string()*) – The comparison mode.

**Devuelve** A standard comparison result

**Tipo del valor devuelto** *int*

Returns -1 if the current *Calends()* object is less than *other*, 0 if they are equal, and 1 if the current object is greater than *other*, using *mode* to select which values to use in the comparison. Valid modes are the same as for *Calends.difference()*, above.

**Calends.contains(*other*)**

**Argumentos**

- **other** (*Calends()*) – The *Calends()* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Checks whether the current *Calends()* object contains all of *other*.

*Calends.overlaps(other)*

**Argumentos**

- **other** (*Calends()*) – The *Calends()* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Checks whether the current *Calends()* object overlaps with *other*.

*Calends.abuts(other)*

**Argumentos**

- **other** (*Calends()*) – The *Calends()* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Checks whether the current *Calends()* object abuts *other* (that is, whether one begins at the same instant the other ends).

*Calends.isSame(other)*

**Argumentos**

- **other** (*Calends()*) – The *Calends()* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Checks whether the current *Calends()* object covers the same span of time as *other*.

*Calends.isShorter(other)*

**Argumentos**

- **other** (*Calends()*) – The *Calends()* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the duration of the current *Calends()* object and *other*.

*Calends.isSameDuration(other)*

**Argumentos**

- **other** (*Calends()*) – The *Calends()* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the duration of the current *Calends()* object and *other*.

`Calends.isLonger(other)`

**Argumentos**

- **other** (`Calends()`) – The `Calends()` object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the duration of the current `Calends()` object and *other*.

`Calends.isBefore(other)`

**Argumentos**

- **other** (`Calends()`) – The `Calends()` object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the entirety of the current `Calends()` object with the start date of *other*.

`Calends.startsBefore(other)`

**Argumentos**

- **other** (`Calends()`) – The `Calends()` object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the start date of the current `Calends()` object with the start date of *other*.

`Calends.endsBefore(other)`

**Argumentos**

- **other** (`Calends()`) – The `Calends()` object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the end date of the current `Calends()` object with the start date of *other*.

`Calends.isDuring(other)`

**Argumentos**

- **other** (`Calends()`) – The `Calends()` object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Checks whether the entirety of the current `Calends()` object lies between the start and end dates of *other*.

`Calends.startsDuring(other)`

**Argumentos**

- **other** (`Calends()`) – The `Calends()` object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Checks whether the start date of the current `Calends()` object lies between the start and end dates of `other`.

`Calends.endsDuring(other)`

**Argumentos**

- **other** (`Calends()`) – The `Calends()` object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Checks whether the end date of the current `Calends()` object lies between the start and end dates of `other`.

`Calends.isAfter(other)`

**Argumentos**

- **other** (`Calends()`) – The `Calends()` object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the entirety of the current `Calends()` object with the end date of `other`.

`Calends.startsAfter(other)`

**Argumentos**

- **other** (`Calends()`) – The `Calends()` object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the start date of the current `Calends()` object with the end date of `other`.

`Calends.endsAfter(other)`

**Argumentos**

- **other** (`Calends()`) – The `Calends()` object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the end date of the current `Calends()` object with the end date of `other`.

## 4.5.7 Export

It is possible to export `Calends()` objects for use later/elsewhere, or to import such values for use in your own code. There are two formats for this purpose: text encoding and JSON encoding. Needless to say, the JSON encoding is considerably more portable, and therefore preferred. Still, both are supported, and as such both are documented here. YMMV.

`Calends.toText()`

**Devuelve** The text encoding of the `Calends()` object.

**Tipo del valor devuelto** string

`Calends.fromText(stamp)`

**Argumentos**

- **stamp** – The text encoded value to import.

**Devuelve** A new *Calends()* object

**Tipo del valor devuelto** *Calends()*

`Calends.toJson()`

**Devuelve** The JSON encoding of the *Calends()* object.

**Tipo del valor devuelto** string

`Calends.fromJson(stamp)`

**Argumentos**

- **stamp** – The JSON encoded value to import.

**Devuelve** A new *Calends()* object

**Tipo del valor devuelto** *Calends()*

For logging, there's also a `toString()` method; we don't recommend using it directly since the output is neither human-readable nor machine-importable.

In addition to the above, there's improved JSON support in JS (go figure) with the following methods:

```
JSON.stringify(objectWithMomentChildren);
```

```
JSON.parse(stored, Calends.reviver);
```

## 4.5.8 Error Handling

**class** `calends.CalendsError()`

A very simple error class, directly extending `Error()`. It is thrown by the library for all encountered errors.

## 4.6 Using Calends in PHP

Calends exposes a very small handful of things for use outside the library itself. One is the *Calends* class, documented here, which should be the only interface users of the library ever need to touch.

Another thing Calends exposes is the *TAITime* class, used to store and manipulate the instants of time which make up a *Calends* instance. The rest are interfaces for extending the library's functionality. These are covered in the *Custom Calendars in PHP* section.

---

**Nota:** *Calends* objects are immutable - all methods return a new *Calends* object where they might otherwise alter the current one. This has the side effect of using more memory to perform manipulations than updating values on an existing object would. It makes many operations safer, though, than mutable objects would allow.

---

**class** Calends\Calends

The main entry point to Calends and its functionality. *Calends* objects can only be created with the *Calends\Calends::create* function, and not directly. *Calends* does implement the Serializable interface, though, so it's safe to serialize/unserialize instances if you want.

**4.6.1 Create**

**static** Calends\Calends::**create**(\$value, \$calendar, \$format)

**Parámetros**

- **\$value** (mixed) – The value to parse the date(s)/time(s) from.
- **\$calendar** (string) – The calendar system to parse the date(s)/time(s) with.
- **\$format** (string) – The format the date(s)/time(s) is/are expected to use.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Lanzamientos** *CalendsException* – when an error occurs

Creates a new *Calends* object, using *\$calendar* to select a calendar system, and *\$format* to parse the contents of *\$value* into the *Calends* object's internal instants. The type of *\$value* can vary based on the calendar system itself, but generally speaking can always be a string.

In any case, the value can always be an associative array, where the keys are any two of **start**, **end**, and **duration**. If all three are provided, **duration** is ignored in favor of calculating it directly.

The calendar system then converts *\$value* to a *TAITime* instant, which the *Calends* object sets to the appropriate internal value.

**4.6.2 Read**

Calends\Calends::**date**(\$calendar, \$format)

**Parámetros**

- **\$calendar** (string) – The calendar system to format the date/time with.
- **\$format** (string) – The format the date/time is expected to be in.

**Devuelve** The start date of the *Calends* object

**Tipo del valor devuelto** string

**Lanzamientos** *CalendsException* – when an error occurs

Retrieves the start date of the *Calends* object as a string. The value is generated by the calendar system given in *\$calendar*, according to the format string in *\$format*.

Calends\Calends::**endDate**(\$calendar, \$format)

**Parámetros**

- **\$calendar** (string) – The calendar system to format the date/time with.
- **\$format** (string) – The format the date/time is expected to be in.

**Devuelve** The end date of the *Calends* object

**Tipo del valor devuelto** string

**Lanzamientos** *CalendsException* – when an error occurs

Retrieves the end date of the *Calends* object as a string. The value is generated by the calendar system given in *\$calendar*, according to the format string in *\$format*.

`Calends\Calends::duration()`

**Devuelve** The duration of the *Calends* object

**Tipo del valor devuelto** string

Retrieves the duration of the *Calends* object as a decimal string. This value will be 0 if the *Calends* object is an instant.

### 4.6.3 Update

`Calends\Calends::withDate($value, $calendar, $format)`

**Parámetros**

- **\$value** (mixed) – The value to parse the date/time from.
- **\$calendar** (string) – The calendar system to parse the date/time with.
- **\$format** (string) – The format the date/time is expected to use.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Lanzamientos** *CalendsException* – when an error occurs

Returns a *Calends* object with a start date based on the current *Calends* object's value. The inputs are the same as for `Calends\Calends::create`, above, except the string → value map option isn't available, since you're already specifically setting the start value explicitly.

`Calends\Calends::withEndDate($value, $calendar, $format)`

**Parámetros**

- **\$value** (mixed) – The value to parse the date/time from.
- **\$calendar** (string) – The calendar system to parse the date/time with.
- **\$format** (string) – The format the date/time is expected to use.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Lanzamientos** *CalendsException* – when an error occurs

Returns a *Calends* object with an end date based on the current *Calends* object's value. The inputs are the same as for `Calends\Calends::create`, above, except the string → value map option isn't available, since you're already specifically setting the end value explicitly.

`Calends\Calends::withDuration($duration, $calendar)`

**Parámetros**

- **\$duration** (string) – The value to parse the new duration from.
- **\$calendar** (string) – The calendar system to parse the date/time with.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Lanzamientos** *CalendsException* – when an error occurs

Returns a *Calends* object with a duration set by adjusting the current *Calends* object's end point, and using its start point as an anchor. The *\$duration* value is interpreted by the calendar system given in *\$calendar*, so is subject to any of its rules.

`Calends\Calends::withDurationFromEnd($duration, $calendar)`

**Parámetros**

- **\$duration** (string) – The value to parse the new duration from.
- **\$calendar** (string) – The calendar system to parse the date/time with.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Lanzamientos** *CalendsException* – when an error occurs

Returns a *Calends* object with a duration set by adjusting the current *Calends* object's start point, and using its end point as an anchor. The *\$duration* value is interpreted by the calendar system given in *\$calendar*, so is subject to any of its rules.

## 4.6.4 Manipulate

`Calends\Calends::add($offset, $calendar)`

**Parámetros**

- **\$offset** (string) – The value to parse the offset from.
- **\$calendar** (string) – The calendar system to parse the date/time with.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Lanzamientos** *CalendsException* – when an error occurs

Increases the end date of the *Calends* object's current value by *\$offset*, as interpreted by the calendar system given in *\$calendar*, and returns a new *Calends* object with the result.

`Calends\Calends::addFromEnd($offset, $calendar)`

**Parámetros**

- **\$offset** (string) – The value to parse the offset from.
- **\$calendar** (string) – The calendar system to parse the date/time with.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Lanzamientos** *CalendsException* – when an error occurs

Increases the start date of the *Calends* object's current value by *\$offset*, as interpreted by the calendar system given in *\$calendar*, and returns a new *Calends* object with the result.

Calends\Calends::subtract(\$offset, \$calendar)

**Parámetros**

- **\$offset** (string) – The value to parse the offset from.
- **\$calendar** (string) – The calendar system to parse the date/time with.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Lanzamientos** *CalendsException* – when an error occurs

Works the same as *add*, except it decreases the start date, rather than increasing it.

Calends\Calends::subtractFromEnd(\$offset, \$calendar)

**Parámetros**

- **\$offset** (string) – The value to parse the offset from.
- **\$calendar** (string) – The calendar system to parse the date/time with.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Lanzamientos** *CalendsException* – when an error occurs

Works the same as *addFromEnd*, except it decreases the end date, rather than increasing it.

Calends\Calends::next(\$offset, \$calendar)

**Parámetros**

- **\$offset** (string) – The value to parse the offset from.
- **\$calendar** (string) – The calendar system to parse the date/time with.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Lanzamientos** *CalendsException* – when an error occurs

Returns a *Calends* object of *\$offset* duration (as interpreted by the calendar system given in *\$calendar*), which abuts the current *Calends* object's value. If *\$offset* is empty, *\$calendar* is ignored, and the current object's duration is used instead.

Calends\Calends::previous(\$offset, \$calendar)

**Parámetros**

- **\$offset** (string) – The value to parse the offset from.
- **\$calendar** (string) – The calendar system to parse the date/time with.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Lanzamientos** *CalendsException* – when an error occurs

Returns a *Calends* object of *\$offset* duration (as interpreted by the calendar system given in *\$calendar*), which abuts the current *Calends* object's value. If *\$offset* is empty, *\$calendar* is ignored, and the current object's duration is used instead.

### 4.6.5 Combine

Calends\Calends::merge(\$c2)

**Parámetros**

- **\$c2** (Calends\Calends) – The *Calends* object to merge.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Lanzamientos** *CalendsException* – when an error occurs

Returns a *Calends* object spanning from the earliest start date to the latest end date between the current *Calends* object and \$c2.

Calends\Calends::intersect(\$c2)

**Parámetros**

- **\$c2** (Calends\Calends) – The *Calends* object to intersect.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Lanzamientos** *CalendsException* – when an error occurs

Returns a *Calends* object spanning the overlap between the current *Calends* object and \$c2. If the current object and \$c2 don't overlap, throws an error.

Calends\Calends::gap(\$c2)

**Parámetros**

- **\$c2** (Calends\Calends) – The *Calends* object to gap.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Lanzamientos** *CalendsException* – when an error occurs

Returns a *Calends* object spanning the gap between the current *Calends* object and \$c2. If the current object and \$c2 overlap (and there is, therefore, no gap to return), throws an error.

### 4.6.6 Compare

Calends\Calends::difference(\$c2, \$mode)

**Parámetros**

- **\$c2** (Calends\Calends) – The *Calends* object to compare.
- **\$mode** (string) – The comparison mode.

**Devuelve** The difference, as a decimal string

**Tipo del valor devuelto** string

Returns the difference of the current *Calends* object minus \$c2, using \$mode to select which values to use in the calculation. Valid \$modes include:

- **start** - use the start date of both the current object and \$c2

- **duration** - use the duration of both the current object and \$c2
- **end** - use the end date of both the current object and \$c2
- **start-end** - use the start of the current object, and the end of \$c2
- **end-start** - use the end of the current object, and the start of \$c2

Calends\Calends::compare(\$c2, \$mode)

**Parámetros**

- **\$c2** (Calends\Calends) – The Calends object to compare.
- **\$mode** (string) – The comparison mode.

**Devuelve** A standard comparison result

**Tipo del valor devuelto** int

Returns -1 if the current Calends object is less than \$c2, 0 if they are equal, and 1 if the current object is greater than \$c2, using \$mode to select which values to use in the comparison. Valid \$modes are the same as for Calends\Calends::difference, above.

Calends\Calends::contains(\$c2)

**Parámetros**

- **\$c2** (Calends\Calends) – The Calends object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Checks whether the current Calends object contains all of \$c2.

Calends\Calends::overlaps(\$c2)

**Parámetros**

- **\$c2** (Calends\Calends) – The Calends object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Checks whether the current Calends object overlaps with \$c2.

Calends\Calends::abuts(\$c2)

**Parámetros**

- **\$c2** (Calends\Calends) – The Calends object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Checks whether the current Calends object abuts \$c2 (that is, whether one begins at the same instant the other ends).

Calends\Calends::isSame(\$c2)

**Parámetros**

- **\$c2** (Calends\Calends) – The Calends object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Checks whether the current *Calends* object covers the same span of time as \$c2.

Calends\Calends::isShorter(\$c2)

**Parámetros**

- **\$c2** (*Calends\Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the duration of the current *Calends* object and \$c2.

Calends\Calends::isSameDuration(\$c2)

**Parámetros**

- **\$c2** (*Calends\Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the duration of the current *Calends* object and \$c2.

Calends\Calends::isLonger(\$c2)

**Parámetros**

- **\$c2** (*Calends\Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the duration of the current *Calends* object and \$c2.

Calends\Calends::isBefore(\$c2)

**Parámetros**

- **\$c2** (*Calends\Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the entirety of the current *Calends* object with the start date of \$c2.

Calends\Calends::startsBefore(\$c2)

**Parámetros**

- **\$c2** (*Calends\Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the start date of the current *Calends* object with the start date of \$c2.

Calends\Calends::endsBefore(\$c2)

**Parámetros**

- **\$c2** (*Calends\Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the end date of the current *Calends* object with the start date of \$c2.

Calends\Calends::isDuring(\$c2)

**Parámetros**

- **\$c2** (*Calends\Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Checks whether the entirety of the current *Calends* object lies between the start and end dates of \$c2.

Calends\Calends::startsDuring(\$c2)

**Parámetros**

- **\$c2** (*Calends\Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Checks whether the start date of the current *Calends* object lies between the start and end dates of \$c2.

Calends\Calends::endsDuring(\$c2)

**Parámetros**

- **\$c2** (*Calends\Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Checks whether the end date of the current *Calends* object lies between the start and end dates of \$c2.

Calends\Calends::isAfter(\$c2)

**Parámetros**

- **\$c2** (*Calends\Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the entirety of the current *Calends* object with the end date of \$c2.

Calends\Calends::startsAfter(\$c2)

**Parámetros**

- **\$c2** (*Calends\Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the start date of the current *Calends* object with the end date of \$c2.

Calends\Calends::endsAfter(\$c2)

**Parámetros**

- **\$c2** (*Calends\Calends*) – The *Calends* object to compare.

**Devuelve** The result of the comparison

**Tipo del valor devuelto** bool

Compares the end date of the current *Calends* object with the end date of \$c2.

## 4.6.7 Export

It's possible to export *Calends* values in a couple of ways. It implements *Serializable* and *JsonSerializable*, as well as the `__toString` method, so the regular mechanisms for each of those are readily available and usable. In addition, it also offers support for JSON-decoding values directly:

**static** Calends\Calends::jsonUnserialize(\$encoded)

**Parámetros**

- **\$encoded** (string) – The JSON-encoded value to import.

**Devuelve** A new *Calends* object

**Tipo del valor devuelto** *Calends*

**Lanzamientos** *CalendsException* – when an error occurs

## 4.6.8 Error Handling

**class** Calends\CalendsException

A very simple exception class, directly extending *Exception*. It is thrown by the library for all encountered errors.



---

## Calendar Systems

---

The systems listed here are the built-in ones. This list is expected to grow significantly over time, as more and more calendar systems are added. But you can also add custom systems to your apps without waiting for us to add them ourselves – instructions for that are in the Custom Calendars section, below.

Throughout these documents, the term `TAITime` is used to refer generically to the `TAI64NARUXTime`, `TAI64Time`, or `TAITime` type. The exact form of this name you'll see most often varies by programming language, and is covered in much more detail in the Custom Calendars section.

### 5.1 The Gregorian Calendar

Supports dates and times in the Gregorian calendar system, the current international standard for communicating dates and times.

**Calendar Name:** `gregorian`

**Supported Input Types:**

- `string`

**Supported Format Strings:**

- Golang `time` package format strings (**RFC1123 layout**)
- C-style `strptime()/strftime()` format strings

**Offsets:**

- `string` with Gregorian time units
  - must be relative times
  - use full words instead of abbreviations for time units (such as `seconds` instead of just `s`)

## 5.2 Julian Day Count

A count of days since BCE 4713 Jan 01 12:00:00 UTC Julian (proleptic). Yes, that's noon. This calendar system is used mostly for astronomy purposes, though there is a modified variant with a narrower scope which counts from midnight instead.

*Calendar Name:* jdc

*Supported Input Types:*

- string
- integer
- arbitrary-precision floating point number of seconds

*Supported Format Strings:*

- full - the full, canonical Day Count
- fullday - the full Day Count, without the fractional time part
- fulltime - just the fractional time part of the full Day Count
- modified - **an abbreviated Day Count, 2400000.5 less than the full (starts at midnight instead of noon)**
- day - the modified Day Count, without the fractional time part
- time - just the fractional time part of the modified Day Count

*Offsets:*

- number of days, as integer or float, via numeric or string types
  - can include fractional days to indicate time

## 5.3 Stardates

Yes, the ones from *Star Trek*<sup>TM</sup>.

The science fiction universe of *Star Trek*<sup>TM</sup> introduced a calendar system which was simultaneously futuristic (for the time) and arbitrary. Over the decades since its initial use on screen, especially with the growing popularity of the franchise, the «stardate» has been analyzed and explored and refined into a number of different variants, each trying to reconcile the arbitrariness of the on-screen system into something consistent and usable.

This calendar system implementation is an attempt to combine all these variants into a single system, using the format parameter to select which variant to use. It was originally ported in 2018 from code by Aaron Chong (2015 version), under provisions of the MIT License. My thanks for Aaron's use of the MIT License on the original code, which allowed me to port it cleanly and legally.

Original source: [http://rinsanity.weebly.com/files/theme/stardate\\_public.js](http://rinsanity.weebly.com/files/theme/stardate_public.js)

*Calendar Name:* stardate

*Supported Input Types:*

- string
- integer
- arbitrary-precision floating point number of seconds

*Supported Format Strings:*

- **main** - **One of the older, more widely-accepted variants.** Alternately called the «issue number style» stardate, it's a combined TOS/TNG variant, and the one used by Google Calendar. It was originally devised by Andrew Main in CE 1994, with revisions made through CE 1997. See <http://starchive.cs.umanitoba.ca/?stardates/> for the full explanation of this variant.
- **kennedy** - In 2006, Richie Kennedy released another combined variant, this one designed to have a single continuous count, more like the Julian Day Count than Main's issue number system.
- **pugh90s** - Steve Pugh devised 2 separate variants, one of them in the 1990s, and the other later on, both focused on the TNG era. They are unique in that, for negative stardates, the fractional part increases in the opposite direction of the expected one. That is, 15129.999999999 would be followed by 15128.000000000 instead of 15129.999999998. The original version used an unadjusted Gregorian year as the basis for the duration of a given range of stardates, meaning that 0.05 units refer to a larger duration of time during a leap year than it would otherwise.
- **pughfixed** - The later of Steve Pugh's systems noted the discrepancy, and opted to adjust the year length value to the actual average length of a Gregorian year, 365.2425 days. This means 0.05 units are always the same duration, but does mean that the Gregorian year doesn't generally start at the same point in consecutive stardate ranges.
- **schmidt** - A joint effort between Andreas Schmidt and Graham Kennedy, this variant only covers TNG-era stardates, and while it can be used proleptically, it ignores the alternate format used prior to TNG. It is also virtually identical to **pugh90s**, but the fractional component increases normally for negative stardates.
- **guide-equiv** - One of five variants proposed by TrekGuide.com, this is the «out-of-universe equivalent» calculation. It isn't intended to be accurate for any use other than personal entertainment.
- **guide-tng** - The second of the five TrekGuide variants, this one is the current scale listed for TNG-era stardates, and is show-accurate (or at least as close to it as feasible with an entirely arbitrary system). Note, however, that it is only accurate for TNG-era dates.
- **guide-tos** - The third variant, then, covers the TOS-era stardates. Again, it is only accurate to the TOS era.
- **guide-oldtng** - The fourth variant is no longer displayed on the TrekGuide site, and was actually pulled from a previous version of the stardates page. It covers the TNG era only, and uses slightly different numbers in its calculations than the current approach - specifically, it assumes Earth years cover 1000 stardates.
- **guide-oldtos** - Representing the very first set of calculations available in archives of the TrekGuide site, the fifth TrekGuide variant assumes that 1000 stardates are one Earth year in the TOS era, and calculates dates based on that assumption. This variant was replaced within seven months of that first archival, after it was noticed that TOS-era stardates don't fit a 1000-stardate model.

---

**Nota:** This calendar system is not yet actually implemented.

---

- **aldrich** - A proof of concept originally written in C#, this variant results in dates very close to those produced by Pugh's and Schmidt's, but uses a more simplified calculation to do it.
- **red-dragon** - A system devised by/for the Red Dragon Inn roleplaying forum site, it uses a fixed ratio of roughly two and three quarters stardates per Earth day. It makes no representations about accuracy outside the context of the site itself.
- **sto-hynes** - John Hynes, creator of the Digital Time site, offers a calculation for STO<sup>1</sup> stardates which appears to be the most accurate variant for those interested in generating those. The system doesn't represent itself as accurate *outside* the game, but is intentionally proleptic.

---

<sup>1</sup> Star Trek™ Online

- `sto-academy` - Based on an online calculator provided by the STO Academy game help site, it is only accurate for stardates within the game, and does not offer to calculate dates for the rest of the franchise.
- `sto-tom` - Another variant intended only to calculate STO stardates, this one was attributed to Major Tom, and hosted as a Wolfram Alpha widget.
- `sto-anthodev` - Another STO variant, hosted on GitHub.

*Offsets:*

- Must be provided as a string in the format "`stardate variant`" or "`variant stardate`".

## 5.4 TAI64 Time

Supports times that are seconds since CE 1970-01-01 00:00:00 TAI Gregorian (plus  $2^{62}$ , when in hexadecimal), as defined at <https://cr.yp.to/libtai/tai64.html> (though this library includes extensions to the formats described there). These values are also used internally, so this calendar system can be used to directly expose the underlying internal values in a manner that allows them to be used elsewhere.

*Calendar Name:* `tai64`

*Supported Input Types:*

- `string`
- `TAITime`

*Supported Format Strings:*

- `decimal` - decimal; full (45 decimal places) resolution; number of seconds since CE 1970-01-01 00:00:00 TAI Gregorian
- `tai64` - hexadecimal; just seconds; TAI64 External Representation
- `tai64n` - hexadecimal; with nanoseconds; TAI64N External Representation
- `tai64na` - hexadecimal; with attoseconds; TAI64NA External Representation
- `tai64nar` - hexadecimal; with rontoseconds; TAI64NAR External Representation
- `tai64naru` - hexadecimal; with udectoseconds; TAI64NARU External Representation
- `tai64narux` - **hexadecimal; with xindectoseconds**; TAI64NARUX External Representation

*Offsets:*

- `TAITime` object
- arbitrary-precision floating point number of seconds
- string with `decimal` format layout (above)

## 5.5 UNIX Time

Supports times that are seconds since CE 1970-01-01 00:00:00 UTC Gregorian, commonly used by computer systems for storing date/time values, internally.

*Calendar Name:* unix

*Supported Input Types:*

- string
- integer
- arbitrary-precision floating point number of seconds

*Supported Format Strings:*

- values are always number of seconds since CE 1970-01-01 00:00:00 UTC Gregorian
  - %d - integer string
  - %f - **floating point string**

*Offsets:*

- number of seconds



---

## Custom Calendars

---

As with every other aspect of Calends, the custom calendar system support uses the same basic flow in every language, with minor variations in each to account for the differences those languages introduce. As with every other aspect of Calends, though, we've opted to document each language's unique approaches separately, so you don't have to do any mental conversions yourself.

---

**Nota:** Custom calendars are considered an advanced feature, so most users won't be using anything detailed here. It can be nice to know how these things work under the hood, though, for those interested in that. Select your language, below, and dig right in!

---

### 6.1 Custom Calendars in Golang

Adding new calendars to Calends is a fairly straightforward process. Implement one interface, or its three methods as standalone functions, and then simply pass them to one of the two registration functions.

#### 6.1.1 Define

The interface in question looks like this:

##### **CalendarDefinition**

```
func (CalendarDefinition) ToInternal(date interface{}, format string) (TAI64NARUXTime, error)
```

##### **Parámetros**

- **date** (interface{}) – The input date. Should support `string` at the very minimum.
- **format** (string) – The format string for parsing the input date.

**Devuelve** The parsed internal timestamp.

**Tipo del valor devuelto** `TAI64NARUXTime`

**Devuelve** Any error that occurs.

**Tipo del valor devuelto** error

Converts an input date/time representation to an internal *TAI64NARUXTime*.

```
func (CalendarDefinition) FromInternal(stamp TAI64NARUXTime, format string) (string, error)
```

**Parámetros**

- **stamp** (*TAI64NARUXTime*) – The internal timestamp value.
- **format** (string) – The format string for formatting the output date.

**Devuelve** The formatted date/time.

**Tipo del valor devuelto** string

**Devuelve** Any error that occurs.

**Tipo del valor devuelto** error

Converts an internal *TAI64NARUXTime* to a date/time string.

```
func (CalendarDefinition) Offset(stamp TAI64NARUXTime, offset interface{ })  
    (TAI64NARUXTime, error)
```

**Parámetros**

- **stamp** (*TAI64NARUXTime*) – The internal timestamp value.
- **offset** (interface{}) – The input offset. Should support string at the very minimum.

**Devuelve** The adjusted internal timestamp.

**Tipo del valor devuelto** *TAI64NARUXTime*

**Devuelve** Any error that occurs.

**Tipo del valor devuelto** error

Adds the given offset to an internal *TAI64NARUXTime*.

## 6.1.2 Registration

### Register

Once it is registered with the library, your calendar system can be used from anywhere in your application. To register a system, pass it to one of the following two functions:

```
func RegisterObject(name string, definition CalendarDefinition, defaultFormat string)
```

**Parámetros**

- **name** (string) – The name to register the calendar system under.
- **definition** (*CalendarDefinition*) – The calendar system itself.
- **defaultFormat** (string) – The default format string.

Registers a calendar system class, storing *definition* as *name*, and saving *defaultFormat* for later use while parsing or formatting.

```
func RegisterElements(name string, toInternal ToInternal, fromInternal FromInternal, offset Offset,
    defaultFormat string)
```

#### Parámetros

- **name** (string) – The name to register the calendar system under.
- **toInternal** ((CalendarDefinition) ToInternal) – The function for parsing dates into internal timestamps.
- **fromInternal** ((CalendarDefinition) FromInternal) – The function for formatting internal timestamps as dates.
- **offset** ((CalendarDefinition) Offset) – The function for adding an offset to internal timestamps.
- **defaultFormat** (string) – The default format string.

Registers a calendar system from its distinct functions. It does this by storing `toInternal`, `fromInternal`, and `offset` as the elements of `name`, and saving `defaultFormat` for later use while parsing or formatting.

### Unregister

```
func Unregister(name string)
```

#### Parámetros

- **name** (string) – The name of the calendar system to remove.

Removes a calendar system from the callback list.

### Check and List

```
func Registered(calendar string) → bool
```

#### Parámetros

- **name** (string) – The calendar system name to check for.

**Devuelve** Whether or not the calendar system is currently registered.

**Tipo del valor devuelto** bool

Returns whether or not a calendar system has been registered, yet.

```
func ListRegistered() → []string
```

**Devuelve** The sorted list of calendar systems currently registered.

**Tipo del valor devuelto** []string

Returns the list of calendar systems currently registered.

### 6.1.3 Types and Values

Now we get to the inner workings that make calendar systems function – even the built-in ones. The majority of the «magic» comes from the [TAI64NARUXTime](#) object itself, as a reliable way of storing the exact instants being calculated, and the only way times are handled by the library itself. A handful of methods provide basic operations that calendar system developers can use to simplify their conversions (adding and subtracting the values of other timestamps, and importing/exporting timestamp values from/to arbitrary-precision floating point `math/big.Float`s, in particular), and a couple of helpers exclusively handle adding and removing UTC leap second offsets. As long as you can convert your dates to/from Unix timestamps in a `string` or `math/big.Float`, the rest is handled entirely by these helpers in the library itself.

#### TAI64NARUXTime

##### Parámetros

- **Seconds** (`int64`) – The number of TAI seconds since CE 1970-01-01 00:00:00 TAI.
- **Nano** (`uint32`) – The first 9 digits of the timestamp’s fractional component.
- **Atto** (`uint32`) – The 10th through 18th digits of the fractional component.
- **Ronto** (`uint32`) – The 19th through 27th digits of the fractional component.
- **Udecto** (`uint32`) – The 28th through 36th digits of the fractional component.
- **Xindecto** (`uint32`) – The 37th through 45th digits of the fractional component.

[TAI64NARUXTime](#) stores a TAI64NARUX instant in a reliable, easily-converted format. Each 9-digit fractional segment is stored in a separate 32-bit integer to preserve its value with a very high degree of accuracy, without having to rely on string parsing or Golang’s `math/big.*` values.

---

#### Nota: TAI vs UTC

You may have noticed that a `TAI64Time` object stores times in TAI seconds, not Unix seconds, with a timezone offset of TAI rather than UTC. This distinction is **very important** as it will affect internal calculations and comparisons to mix the two up. TAI time is very similar to Unix time (itself based on UTC time), with one major difference. While Unix/UTC seconds include the insertion and removal of «leap seconds» to keep the solar zenith at local noon (which is useful for day-to-day living and planning), TAI seconds are a continuous count, unconcerned with dates whatsoever. Indeed, the only reason a date was given in the description above was to make it easier for human readers to know exactly when 0 TAI took place.

In other words, once you have a Unix timestamp of your instant calculated, be sure to convert it using [UTCtoTAI](#) before returning the result to the rest of the library. And then, of course, you’ll also need to convert instants from the library back using [TAItoUTC](#) before generating outputs.

---

`func (TAI64NARUXTime) Add(z TAI64NARUXTime) → TAI64NARUXTime`

##### Parámetros

- **z** ([TAI64NARUXTime](#)) – The timestamp to add to the current one.

**Devuelve** The sum of the two timestamps.

**Tipo del valor devuelto** [TAI64NARUXTime](#)

Calculates the sum of two [TAI64NARUXTime](#) values.

`func (TAI64NARUXTime) Sub(z TAI64NARUXTime) → TAI64NARUXTime`

##### Parámetros

- **z** ([TAI64NARUXTime](#)) – The timestamp to subtract from the current one.

**Devuelve** The difference of the two timestamps.

**Tipo del valor devuelto** *TAI64NARUXTime*

Calculates the difference of two *TAI64NARUXTime* values.

```
func (TAI64NARUXTime) String() → string
```

**Devuelve** The decimal string representation of the current timestamp.

**Tipo del valor devuelto** string

Returns the decimal string representation of the *TAI64NARUXTime* value.

```
func (TAI64NARUXTime) HexString() → string
```

**Devuelve** The hexadecimal string representation of the current timestamp.

**Tipo del valor devuelto** string

Returns the hexadecimal string representation of the *TAI64NARUXTime* value.

```
func (TAI64NARUXTime) Float() → Float
```

**Devuelve** The arbitrary-precision floating point representation of the current timestamp.

**Tipo del valor devuelto** math/big.(\*Float)

Returns the math/big.(\*Float) representation of the *TAI64NARUXTime* value.

```
func (TAI64NARUXTime) MarshalText() ([]byte, error)
```

**Devuelve** A byte slice containing the marshalled text.

**Tipo del valor devuelto** []byte

**Devuelve** Any error that occurs.

**Tipo del valor devuelto** error

Implements the encoding.TextMarshaler interface.

```
func (TAI64NARUXTime) UnmarshalText(in []byte) → error
```

**Parámetros**

- **in** ([]byte) – A byte slice containing the marshalled text.

**Devuelve** Any error that occurs.

**Tipo del valor devuelto** error

Implements the encoding.TextUnmarshaler interface.

```
func (TAI64NARUXTime) MarshalBinary() ([]byte, error)
```

**Devuelve** A byte slice containing the marshalled binary data.

**Tipo del valor devuelto** []byte

**Devuelve** Any error that occurs.

**Tipo del valor devuelto** error

Implements the encoding.BinaryMarshaler interface.

```
func (TAI64NARUXTime) UnmarshalBinary(in []byte) → error
```

**Parámetros**

- **in** ([]byte) – A byte slice containing the marshalled binary data.

**Devuelve** Any error that occurs.

**Tipo del valor devuelto** error

Implements the encoding.BinaryUnmarshaler interface.

## 6.1.4 Helpers

```
func TAI64NARUXTimeFromDecimalString(in string) → TAI64NARUXTime
```

**Parámetros**

- **in** (string) – The decimal string representation of a timestamp to calculate.

**Devuelve** The calculated timestamp.

**Tipo del valor devuelto** [TAI64NARUXTime](#)

Calculates a [TAI64NARUXTime](#) from its decimal string representation.

```
func TAI64NARUXTimeFromHexString(in string) → TAI64NARUXTime
```

**Parámetros**

- **in** (string) – The hexadecimal string representation of a timestamp to calculate.

**Devuelve** The calculated timestamp.

**Tipo del valor devuelto** [TAI64NARUXTime](#)

Calculates a [TAI64NARUXTime](#) from its hexadecimal string representation.

```
func TAI64NARUXTimeFromFloat(in Float) → TAI64NARUXTime
```

**Parámetros**

- **in** (math/big.Float) – The arbitrary-precision floating point representation of a timestamp to calculate.

**Devuelve** The calculated timestamp.

**Tipo del valor devuelto** [TAI64NARUXTime](#)

Calculates a [TAI64NARUXTime](#) from its math/big.Float representation.

```
func UTCtoTAI(utc TAI64NARUXTime) → TAI64NARUXTime
```

**Parámetros**

- **utc** ([TAI64NARUXTime](#)) – The timestamp to remove the UTC offset from.

**Devuelve** The calculated timestamp.

**Tipo del valor devuelto** [TAI64NARUXTime](#)

Removes the UTC leap second offset from a TAI64NARUXTime value.

```
func TAItoUTC(tai TAI64NARUXTime) → TAI64NARUXTime
```

**Parámetros**

- **tai** (*TAI64NARUXTime*) – The timestamp to add the UTC offset to.

**Devuelve** The calculated timestamp.

**Tipo del valor devuelto** *TAI64NARUXTime*

Adds the UTC leap second offset to a TAI64NARUXTime value.

## 6.1.5 Errors

### **ErrUnsupportedInput**

Used to indicate that the input date/time weren't recognized by the calendar system, or that the data type is incorrect.

### **ErrInvalidFormat**

Indicates that the `format` string isn't supported by the calendar system.

```
func ErrUnknownCalendar(calendar string) → error
```

**Parámetros**

- **in** (string) – The name of the unknown calendar system.

**Devuelve** Any error that occurs.

**Tipo del valor devuelto** `error`

Generates a «calendar not registered» error, including the calendar's actual name in the error message.

## 6.2 Custom Calendars in C/C++

Adding new calendars to Calends is a fairly straightforward process. Implement a handful of functions, and then simply pass them to the registration function.

### 6.2.1 Define

The functions in question look like this:

```
TAI64Time Calends_calendar_to_internal_string(char *calendar, char *date, char *format)
```

```
TAI64Time Calends_calendar_to_internal_long_long(char *calendar, long long int date, char *format)
```

```
TAI64Time Calends_calendar_to_internal_double(char *calendar, double date, char *format)
```

```
TAI64Time Calends_calendar_to_internal_tai(char *calendar, TAI64Time date)
```

**Parámetros**

- **calendar** (char\*) – The name of the target calendar system.
- **date** (char\* or long long int or double or *TAI64Time*) – The input date.
- **format** (char\*) – The format string for parsing the input date.

**Devuelve** The parsed internal timestamp.

**Tipo del valor devuelto** *TAI64Time*

Converts an input date/time representation to an internal *TAI64Time*.

char \*Calends\_calendar\_from\_internal(char \*calendar, *TAI64Time* stamp, char \*format)

**Parámetros**

- **calendar** (char\*) – The name of the target calendar system.
- **stamp** (*TAI64Time*) – The internal timestamp value.
- **format** (char\*) – The format string for formatting the output date.

**Devuelve** The formatted date/time.

**Tipo del valor devuelto** char\*

Converts an internal *TAI64Time* to a date/time string.

*TAI64Time* Calends\_calendar\_offset\_string(char \*calendar, *TAI64Time* stamp, char \*offset)

*TAI64Time* Calends\_calendar\_offset\_long\_long(char \*calendar, *TAI64Time* stamp, long long int offset)

*TAI64Time* Calends\_calendar\_offset\_double(char \*calendar, *TAI64Time* stamp, double offset)

*TAI64Time* Calends\_calendar\_offset\_tai(char \*calendar, *TAI64Time* stamp, *TAI64Time* offset)

**Parámetros**

- **calendar** (char\*) – The name of the target calendar system.
- **stamp** (*TAI64Time*) – The internal timestamp value.
- **offset** (char\* or long long int or double or *TAI64Time*) – The input offset.

**Devuelve** The adjusted internal timestamp.

**Tipo del valor devuelto** *TAI64Time*

Adds the given offset to an internal *TAI64Time*.

## 6.2.2 Registration

### Register

Once it is registered with the library, your calendar system can be used from anywhere in your application. To register a system, pass it to the following function:

```
void Calends_calendar_register(char* name, char* defaultFormat,  
Calends_calendar_to_internal_string() to_internal_string,  
Calends_calendar_to_internal_long_long() to_internal_long_long,  
Calends_calendar_to_internal_double() to_internal_double,  
Calends_calendar_to_internal_tai() to_internal_tai,  
Calends_calendar_from_internal() from_internal,  
Calends_calendar_offset_string() offset_string,  
Calends_calendar_offset_long_long() offset_long_long,  
Calends_calendar_offset_double() offset_double, Calends_calendar_offset_tai() offset_tai)
```

**Parámetros**

- **name** (char\*) – The name to register the calendar system under.
- **defaultFormat** (char\*) – The default format string.
- **to\_internal\_string** (*Calends\_calendar\_to\_internal\_string()*) – The calendar parser, for char\* input.

- **to\_internal\_long\_long** (*Calends\_calendar\_to\_internal\_long\_long()*) – The calendar parser, for long long int input.
- **to\_internal\_double** (*Calends\_calendar\_to\_internal\_double()*) – The calendar parser, for double input.
- **to\_internal\_tai** (*Calends\_calendar\_to\_internal\_tai()*) – The calendar parser, for *TAI64Time* input.
- **from\_internal** (*Calends\_calendar\_from\_internal()*) – The calendar formatter.
- **offset\_string** (*Calends\_calendar\_offset\_string()*) – The calendar offset calculator, for char\* input.
- **offset\_long\_long** (*Calends\_calendar\_offset\_long\_long()*) – The calendar offset calculator, for long long int input.
- **offset\_double** (*Calends\_calendar\_offset\_double()*) – The calendar offset calculator, for double input.
- **offset\_tai** (*Calends\_calendar\_offset\_tai()*) – The calendar offset calculator, for *TAI64Time* input.

Registers a calendar system class, storing the collected functions as name, and saving defaultFormat for later use while parsing or formatting.

## Unregister

void **Calends\_calendar\_unregister**(char \*name)

### Parámetros

- **name** (char\*) – The name of the calendar system to remove.

Removes a calendar system from the callback list.

## Check and List

bool **Calends\_calendar\_registered**(char \*name)

### Parámetros

- **name** (char\*) – The calendar system name to check for.

**Devuelve** Whether or not the calendar system is currently registered.

**Tipo del valor devuelto** bool

Returns whether or not a calendar system has been registered, yet.

char \***Calends\_calendar\_list\_registered**()

**Devuelve** The sorted list of calendar systems currently registered.

**Tipo del valor devuelto** char\*

Returns the list of calendar systems currently registered.

### 6.2.3 Types and Values

Now we get to the inner workings that make calendar systems function – even the built-in ones. The majority of the «magic» comes from the `TAI64Time` struct itself, as a reliable way of storing the exact instants being calculated, and the only way times are handled by the library itself. A handful of functions provide basic operations that calendar system developers can use to simplify their conversions (adding and subtracting the values of other timestamps, and importing/exporting timestamp values from/to other types, in particular), and a couple of helpers exclusively handle adding and removing UTC leap second offsets. As long as you can convert your dates to/from Unix timestamps in a `char*`, `long long int`, or `double`, the rest is handled entirely by these helpers in the library itself.

#### type `TAI64Time`

Stores a `TAI64NARUX` instant in a reliable, easy-converted format. Each 9-digit fractional segment is stored in a separate 32-bit integer to preserve its value with a very high degree of accuracy, without having to rely on string parsing or external arbitrary-precision math libraries.

#### `long long int seconds`

The number of TAI seconds since CE 1970-01-01 00:00:00 TAI

---

#### **Nota:** TAI vs UTC

You may have noticed that a `TAI64Time` object stores times in `TAI seconds`, not `Unix seconds`, with a timezone offset of TAI rather than UTC. This distinction is **very important** as it will affect internal calculations and comparisons to mix the two up. TAI time is very similar to Unix time (itself based on UTC time), with one major difference. While Unix/UTC seconds include the insertion and removal of «leap seconds» to keep the solar zenith at local noon (which is useful for day-to-day living and planning), TAI seconds are a continuous count, unconcerned with dates whatsoever. Indeed, the only reason a date was given in the description above was to make it easier for human readers to know exactly when 0 TAI took place.

In other words, once you have a Unix timestamp of your instant calculated, be sure to convert it using `TAI64Time_utc_to_tai()` before returning the result to the rest of the library. And then, of course, you'll also need to convert instants from the library back using `TAI64Time_tai_to_utc()` before generating outputs.

---

#### `unsigned int nano`

Nanoseconds since the given second

#### `unsigned int atto`

Attoseconds since the given nanosecond

#### `unsigned int ronto`

Rontoseconds since the given attosecond

#### `unsigned int udecto`

Udectoseconds since the given rontosecond

#### `unsigned int xindecto`

Xindectoseconds since the given udectosecond

#### `unsigned int padding`

Unused, except to round the value out to the nearest 64 bits

## 6.2.4 Calculations

*TAI64Time* **TAI64Time\_add**(*TAI64Time* t, *TAI64Time* z)

**Parámetros**

- **t** (*TAI64Time*) – The current timestamp.
- **z** (*TAI64Time*) – The timestamp to add to the current one.

**Devuelve** The sum of the two timestamps.

**Tipo del valor devuelto** *TAI64Time*

Calculates the sum of two *TAI64Time* values.

*TAI64Time* **TAI64Time\_sub**(*TAI64Time* t, *TAI64Time* z)

**Parámetros**

- **t** (*TAI64Time*) – The current timestamp.
- **z** (*TAI64Time*) – The timestamp to subtract from the current one.

**Devuelve** The difference of the two timestamps.

**Tipo del valor devuelto** *TAI64Time*

Calculates the difference of two *TAI64Time* values.

## 6.2.5 Export

char \***TAI64Time\_string**(*TAI64Time* t)

**Parámetros**

- **t** (*TAI64Time*) – The current timestamp.

**Devuelve** The decimal string representation of the current timestamp.

**Tipo del valor devuelto** char\*

Returns the decimal string representation of a *TAI64Time* value.

*TAI64Time* **TAI64Time\_from\_string**(char \*in)

**Parámetros**

- **in** (char\*) – The decimal string representation of a timestamp to calculate.

**Devuelve** The calculated timestamp.

**Tipo del valor devuelto** *TAI64Time*

Calculates a *TAI64Time* from its decimal string representation.

char \***TAI64Time\_hex\_string**(*TAI64Time* t)

**Parámetros**

- **t** (*TAI64Time*) – The current timestamp.

**Devuelve** The hexadecimal string representation of the current timestamp.

**Tipo del valor devuelto** char\*

Returns the hexadecimal string representation of a *TAI64Time* value.

*TAI64Time* **TAI64Time\_from\_hex\_string**(char \*in)

**Parámetros**

- **in** (char\*) – The hexadecimal string representation of a timestamp to calculate.

**Devuelve** The calculated timestamp.

**Tipo del valor devuelto** *TAI64Time*

Calculates a *TAI64Time* from its hexadecimal string representation.

double **TAI64Time\_double**(*TAI64Time* t)

**Parámetros**

- **t** (*TAI64Time*) – The current timestamp.

**Devuelve** The arbitrary-precision floating point representation of the current timestamp.

**Tipo del valor devuelto** double

Returns the double representation of a *TAI64Time* value.

*TAI64Time* **TAI64Time\_from\_double**(double in)

**Parámetros**

- **in** (double) – The arbitrary-precision floating point representation of a timestamp to calculate.

**Devuelve** The calculated timestamp.

**Tipo del valor devuelto** *TAI64Time*

Calculates a *TAI64Time* from its double representation.

char \***TAI64Time\_encode\_text**(*TAI64Time* t)

**Parámetros**

- **t** (*TAI64Time*) – The current timestamp.

**Devuelve** A string containing the encoded text.

**Tipo del valor devuelto** char\*

Encodes a *TAI64Time* value as text.

*TAI64Time* **TAI64Time\_decode\_text**(char \*in)

**Parámetros**

- **in** (char\*) – A string containing the encoded text.

**Devuelve** The decoded timestamp.

**Tipo del valor devuelto** *TAI64Time*

Decodes a *TAI64Time* value from text.

void \***TAI64Time\_encode\_binary**(*TAI64Time* t, int \*len)

**Parámetros**

- **t** (*TAI64Time*) – The current timestamp.
- **len** (int\*) – Will return the length of the binary data.

**Devuelve** A pointer to the encoded binary data stream.

**Tipo del valor devuelto** `void*`

Encodes a *TAI64Time* value as a binary data stream.

*TAI64Time* **TAI64Time\_decode\_binary**(void \*in, int len)

**Parámetros**

- **in** (void\*) – A pointer to the encoded binary data stream.
- **len** (int) – The length of the binary data.

**Devuelve** The decoded timestamp.

**Tipo del valor devuelto** *TAI64Time*

Decodes a *TAI64Time* value from a binary data stream.

## 6.2.6 Helpers

*TAI64Time* **TAI64Time\_utc\_to\_tai**(*TAI64Time* utc)

**Parámetros**

- **utc** (*TAI64Time*) – The timestamp to remove the UTC offset from.

**Devuelve** The calculated timestamp.

**Tipo del valor devuelto** *TAI64Time*

Removes the UTC leap second offset from a *TAI64Time* value.

*TAI64Time* **TAI64Time\_tai\_to\_utc**(*TAI64Time* tai)

**Parámetros**

- **tai** (*TAI64Time*) – The timestamp to add the UTC offset to.

**Devuelve** The calculated timestamp.

**Tipo del valor devuelto** *TAI64Time*

Adds the UTC leap second offset to a *TAI64Time* value.

## 6.3 Custom Calendars in Dart

Adding new calendars to Calends is a fairly straightforward process. Extend the *CalendarDefinition* abstract class, and implement two getters and three methods. Then, simply construct an instance of your calendar system, and Calends will do the rest.

### 6.3.1 Define

Extend the *CalendarDefinition* class, implementing the following methods:

**class** `calends.CalendarDefinition`

`CalendarDefinition.get name()`

**Devuelve** The name of the calendar system.

**Tipo del valor devuelto** `String`

`CalendarDefinition.get defaultFormat()`

**Devuelve** The default format of the calendar system.

**Tipo del valor devuelto** `String`

`CalendarDefinition.toInternal(dynamic stamp, String format)`

**Argumentos**

- **stamp** (`dynamic`) – The input stamp. Should support strings at the very minimum.
- **format** (`String`) – The format string for parsing the input stamp.

**Devuelve** The parsed internal timestamp.

**Tipo del valor devuelto** *TAI64Time*

**Lanzamientos** *CalendsException()* – when an error occurs

Converts an input date/time representation to an internal *TAI64Time*.

`CalendarDefinition.fromInternal(TAI64Time stamp, String format)`

**Argumentos**

- **stamp** (*TAI64Time*) – The internal timestamp value.
- **format** (`String`) – The format string for formatting the output date.

**Devuelve** The formatted date/time.

**Tipo del valor devuelto** `String`

**Lanzamientos** *CalendsException()* – when an error occurs

Converts an internal *TAI64Time* to a date/time string.

`CalendarDefinition.offset(TAI64Time stamp, dynamic offset)`

**Argumentos**

- **stamp** (*TAI64Time*) – The internal timestamp value.
- **offset** (`dynamic`) – The input offset. Should support strings at the very minimum.

**Devuelve** The adjusted internal timestamp.

**Tipo del valor devuelto** *TAI64Time*

**Lanzamientos** *CalendsException()* – when an error occurs

Adds the given offset to an internal *TAI64Time*.

## 6.3.2 Registration

### Register

Once it is registered with the library, your calendar system can be used from anywhere in your application:

`CalendarDefinition.register()`

Adds a calendar system to the callback list.

### Unregister

When you are done with a calendar system, it is best practice to free up resources by unregistering it:

`CalendarDefinition.unregister()`

Removes a calendar system from the callback list.

### Check and List

`CalendarDefinition.isRegistered()`

**Devuelve** Whether or not the calendar system is currently registered.

**Tipo del valor devuelto** `bool`

Returns whether or not a calendar system has been registered, yet.

`CalendarDefinition.listRegistered()`

**Devuelve** The sorted list of calendar systems currently registered.

**Tipo del valor devuelto** `List<String>`

Returns the list of calendar systems currently registered.

## 6.3.3 Types and Values

Now we get to the inner workings that make calendar systems function – even the built-in ones. The majority of the «magic» comes from the `TAI64Time` object itself, as a reliable way of storing the exact instants being calculated, and the only way times are handled by the library itself. A handful of methods provide basic operations that calendar system developers can use to simplify their conversions (adding and subtracting the values of other timestamps, and importing/exporting timestamp values from/to string and numeric types, in particular), and a couple of helpers exclusively handle adding and removing UTC leap second offsets. As long as you can convert your dates to/from Unix timestamps in a string or numeric type, the rest is handled entirely by these helpers in the library itself.

**class** `calends.TAI64Time`

`TAI64Time` stores a TAI64NARUX instant in a reliable, easily-converted format. Each 9-digit fractional segment is stored in a separate 32-bit integer to preserve its value with a very high degree of accuracy, without having to rely on string parsing or external arbitrary-precision mathematics libraries.

`TAI64Time.Seconds`

The number of TAI seconds since CE 1970-01-01 00:00:00 TAI.

---

**Nota:** TAI vs UTC

You may have noticed that a `TAI64Time` object stores times in `TAI` seconds, not `Unix` seconds, with a timezone offset of `TAI` rather than `UTC`. This distinction is **very important** as it will affect internal calculations and comparisons to mix the two up. `TAI` time is very similar to `Unix` time (itself based on `UTC` time), with one major difference. While `Unix/UTC` seconds include the insertion and removal of «leap seconds» to keep the solar zenith at local noon (which is useful for day-to-day living and planning), `TAI` seconds are a continuous count, unconcerned with dates whatsoever. Indeed, the only reason a date was given in the description above was to make it easier for human readers to know exactly when `0` `TAI` took place.

In other words, once you have a `Unix` timestamp of your instant calculated, be sure to convert it using `utcToTai()` before returning the result to the rest of the library. And then, of course, you'll also need to convert instants from the library back using `taiToUtc()` before generating outputs.

---

**TAI64Time.Nano**

The first 9 digits of the timestamp's fractional component.

**TAI64Time.Atto**

The 10th through 18th digits of the fractional component.

**TAI64Time.Ronto**

The 19th through 27th digits of the fractional component.

**TAI64Time.Udecto**

The 28th through 36th digits of the fractional component.

**TAI64Time.Xindecto**

The 37th through 45th digits of the fractional component.

**TAI64Time.add(TAI64Time z)****Argumentos**

- **z** (*TAI64Time*) – The timestamp to add to the current one.

**Devuelve** The sum of the two timestamps.

**Tipo del valor devuelto** *TAI64Time*

Calculates the sum of two *TAI64Time* values.

**TAI64Time.sub(TAI64Time z)****Argumentos**

- **z** (*TAI64Time*) – The timestamp to subtract from the current one.

**Devuelve** The difference of the two timestamps.

**Tipo del valor devuelto** *TAI64Time*

Calculates the difference of two *TAI64Time* values.

**TAI64Time.toTAI64String()**

**Devuelve** The decimal string representation of the current timestamp.

**Tipo del valor devuelto** `String`

Returns the decimal string representation of the *TAI64Time* value.

**TAI64Time.fromTAI64String(String in)****Argumentos**

- **in** (string()) – The decimal string representation of a timestamp to calculate.

**Devuelve** The calculated timestamp.

**Tipo del valor devuelto** *TAI64Time*

Calculates a *TAI64Time* from its decimal string representation.

`TAI64Time.toHex()`

**Devuelve** The hexadecimal string representation of the current timestamp.

**Tipo del valor devuelto** String

Returns the hexadecimal string representation of the *TAI64Time* value.

`TAI64Time.fromHex(string in)`

**Argumentos**

- **in** (string()) – The hexadecimal string representation of a timestamp to calculate.

**Devuelve** The calculated timestamp.

**Tipo del valor devuelto** *TAI64Time*

Calculates a *TAI64Time* from its hexadecimal string representation.

`TAI64Time.toDouble()`

**Devuelve** The floating point representation of the current timestamp.

**Tipo del valor devuelto** double

Returns the double representation of the *TAI64Time* value.

`TAI64Time.fromDouble(double in)`

**Argumentos**

- **in** (double) – The floating point representation of a timestamp to calculate.

**Devuelve** The calculated timestamp.

**Tipo del valor devuelto** *TAI64Time*

Calculates a *TAI64Time* from its double representation.

`TAI64Time.utcToTai()`

**Devuelve** The calculated timestamp.

**Tipo del valor devuelto** *TAI64Time*

Removes the UTC leap second offset from a *TAI64Time* value. Used when converting from Unix time to TAI time.

`TAI64Time.taiToUtc()`

**Devuelve** The calculated timestamp.

**Tipo del valor devuelto** *TAI64Time*

Adds the UTC leap second offset to a *TAI64Time* value. Used when converting from TAI time to Unix time.

## 6.4 Custom Calendars in JS/WASM

Adding new calendars to Calends is a fairly straightforward process. Extend the `CalendarDefinition()` abstract class, and implement three methods. Then, simply construct an instance of your calendar system, and Calends will do the rest.

### 6.4.1 Define

Extend the `CalendarDefinition()` class, implementing the following methods:

**class** `calends.CalendarDefinition()`

`CalendarDefinition.name`

The name of the calendar system. Can be static or set in the `constructor()`.

`CalendarDefinition.defaultFormat`

The default/fallback format for the calendar system. Can be static or set in the `constructor()`.

`CalendarDefinition.constructor()`

This can do anything you like.

`CalendarDefinition.toInternal(stamp, format)`

**Argumentos**

- **stamp** (any) – The input. Should support strings at the very minimum.
- **format** (string) – The format string for parsing the input date.

**Devuelve** The parsed internal timestamp.

**Tipo del valor devuelto** `TAI64Time()`

**Lanzamientos** `CalendsException()` – when an error occurs

Converts an input date/time representation to an internal `TAI64Time()`.

`CalendarDefinition.fromInternal(instant, format)`

**Argumentos**

- **instant** (`TAI64Time()`) – The internal timestamp value.
- **format** (string) – The format string for formatting the output date.

**Devuelve** The formatted date/time.

**Tipo del valor devuelto** string

**Lanzamientos** `CalendsException()` – when an error occurs

Converts an internal `TAI64Time()` to a date/time string.

`CalendarDefinition.offset(instant, offset)`

**Argumentos**

- **instant** (`TAI64Time()`) – The internal timestamp value.
- **offset** (any) – The input offset. Should support strings at the very minimum.

**Devuelve** The adjusted internal timestamp.

**Tipo del valor devuelto** `TAI64Time()`

**Lanzamientos** `CalendsException()` – when an error occurs

Adds the given offset to an internal `TAI64Time()`.

## 6.4.2 Registration

### Register

Once it is registered with the library, your calendar system can be used from anywhere in your application. To register a system, simply call `register()` on an object of your new class:

`CalendarDefinition.register()`

Registers a calendar system instance with the internal Calends library.

### Unregister

The way to unregister a calendar system is to do so manually, using the instance you created to register it with in the first place:

`CalendarDefinition.unregister()`

Removes a calendar system from the callback list.

### Check and List

`CalendarDefinition.isRegistered()`

**Devuelve** Whether or not the calendar system is currently registered.

**Tipo del valor devuelto** `bool`

Returns whether or not a calendar system has been registered, yet.

`CalendarDefinition.registered()`

**Devuelve** The sorted list of calendar systems currently registered.

**Tipo del valor devuelto** `[string]`

Returns the list of calendar systems currently registered.

## 6.4.3 Types and Values

Now we get to the inner workings that make calendar systems function – even the built-in ones. The majority of the «magic» comes from the `TAI64Time()` object itself, as a reliable way of storing the exact instants being calculated, and the only way times are handled by the library itself. A handful of methods provide basic operations that calendar system developers can use to simplify their conversions (adding and subtracting the values of other timestamps, and importing/exporting timestamp values from/to string and numeric types, in particular), and a couple of helpers exclusively handle adding and removing UTC leap second offsets. As long as you can convert your dates to/from Unix timestamps in a string or numeric type, the rest is handled entirely by these helpers in the library itself.

**class** `calends.TAI64Time()`

`TAI64Time()` stores a TAI64NARUX instant in a reliable, easily-converted format. Each 9-digit fractional segment is stored in a separate 32-bit integer to preserve its value with a very high degree of accuracy, without having to rely on string parsing or external arbitrary-precision mathematics libraries.

**TAI64Time.seconds**

The number of TAI seconds since CE 1970-01-01 00:00:00 TAI. Should be an integer value.

---

**Nota:** TAI vs UTC

You may have noticed that a `TAI64Time` object stores times in TAI seconds, not Unix seconds, with a timezone offset of TAI rather than UTC. This distinction is **very important** as it will affect internal calculations and comparisons to mix the two up. TAI time is very similar to Unix time (itself based on UTC time), with one major difference. While Unix/UTC seconds include the insertion and removal of «leap seconds» to keep the solar zenith at local noon (which is useful for day-to-day living and planning), TAI seconds are a continuous count, unconcerned with dates whatsoever. Indeed, the only reason a date was given in the description above was to make it easier for human readers to know exactly when 0 TAI took place.

In other words, once you have a Unix timestamp of your instant calculated, be sure to convert it using [`fromUTC\(\)`](#) before returning the result to the rest of the library. And then, of course, you'll also need to convert instants from the library back using [`toUTC\(\)`](#) before generating outputs.

---

**TAI64Time.nano**

The first 9 digits of the timestamp's fractional component.

**TAI64Time.atto**

The 10th through 18th digits of the fractional component.

**TAI64Time.ronto**

The 19th through 27th digits of the fractional component.

**TAI64Time.udecto**

The 28th through 36th digits of the fractional component.

**TAI64Time.xindecto**

The 37th through 45th digits of the fractional component.

**TAI64Time.add(z)****Argumentos**

- **z** ([`TAI64Time\(\)`](#)) – The timestamp to add to the current one.

**Devuelve** The sum of the two timestamps.

**Tipo del valor devuelto** [`TAI64Time\(\)`](#)

Calculates the sum of two [`TAI64Time\(\)`](#) values.

**TAI64Time.sub(z)****Argumentos**

- **z** ([`TAI64Time\(\)`](#)) – The timestamp to subtract from the current one.

**Devuelve** The difference of the two timestamps.

**Tipo del valor devuelto** [`TAI64Time\(\)`](#)

Calculates the difference of two [`TAI64Time\(\)`](#) values.

**TAI64Time.toString()**

**Devuelve** The decimal string representation of the current timestamp.

**Tipo del valor devuelto** `string`

Returns the decimal string representation of the `TAI64Time()` value.

`TAI64Time.fromString(in)`

**Argumentos**

- **in** (string) – The decimal string representation of a timestamp to calculate.

**Devuelve** The calculated timestamp.

**Tipo del valor devuelto** `TAI64Time()`

Calculates a `TAI64Time()` from its decimal string representation.

`TAI64Time.toHex()`

**Devuelve** The hexadecimal string representation of the current timestamp.

**Tipo del valor devuelto** string

Returns the hexadecimal string representation of the `TAI64Time()` value.

`TAI64Time.fromHex(in)`

**Argumentos**

- **in** (string) – The hexadecimal string representation of a timestamp to calculate.

**Devuelve** The calculated timestamp.

**Tipo del valor devuelto** `TAI64Time()`

Calculates a `TAI64Time()` from its hexadecimal string representation.

`TAI64Time.toNumber()`

**Devuelve** The numeric representation of the current timestamp.

**Tipo del valor devuelto** number

Returns the number representation of the `TAI64Time()` value.

`TAI64Time.fromNumber(in)`

**Argumentos**

- **in** (number) – The arbitrary-precision floating point representation of a timestamp to calculate.

**Devuelve** The calculated timestamp.

**Tipo del valor devuelto** `TAI64Time()`

Calculates a `TAI64Time()` from its numeric representation.

`TAI64Time.fromUTC()`

**Devuelve** The calculated timestamp.

**Tipo del valor devuelto** `TAI64Time()`

Removes the UTC leap second offset from a `TAI64Time` value.

`TAI64Time.toUTC()`

**Devuelve** The calculated timestamp.

**Tipo del valor devuelto** `TAI64Time()`

Adds the UTC leap second offset to a `TAI64Time` value.

## 6.5 Custom Calendars in PHP

Adding new calendars to Calends is a fairly straightforward process. Extend the *CalendarDefinition* abstract class, and implement three methods. Then, simply construct an instance of your calendar system, and Calends will do the rest.

### 6.5.1 Define

Extend the *CalendarDefinition* class, implementing the following methods:

**class** Calends\CalendarDefinition

**toInternal**(*mixed* \$date, *string* \$format) → TAITime

**Parámetros**

- **\$date** (*mixed*) – The input date. Should support strings at the very minimum.
- **\$format** (*string*) – The format string for parsing the input date.

**Devuelve** The parsed internal timestamp.

**Tipo del valor devuelto** *TAITime*

**Lanzamientos** *CalendsException* – when an error occurs

Converts an input date/time representation to an internal *TAITime*.

**fromInternal**(*TAITime* \$stamp, *string* \$format) → *string*

**Parámetros**

- **\$stamp** (*TAITime*) – The internal timestamp value.
- **\$format** (*string*) – The format string for formatting the output date.

**Devuelve** The formatted date/time.

**Tipo del valor devuelto** *string*

**Lanzamientos** *CalendsException* – when an error occurs

Converts an internal *TAITime* to a date/time string.

**offset**(*TAITime* \$stamp, *mixed* \$offset) → TAITime

**Parámetros**

- **\$stamp** (*TAITime*) – The internal timestamp value.
- **\$offset** (*mixed*) – The input offset. Should support strings at the very minimum.

**Devuelve** The adjusted internal timestamp.

**Tipo del valor devuelto** *TAITime*

**Lanzamientos** *CalendsException* – when an error occurs

Adds the given offset to an internal *TAITime*.

## 6.5.2 Registration

### Register

Once it is registered with the library, your calendar system can be used from anywhere in your application. To register a system, simply construct an instance:

```
$customCalendars[] = new MyCalendarSystem('example', 'yyyy-mm-dd@HH-MM-SS');
```

The first argument is the calendar system's name. The second is a default format string which will be passed to your calendar system whenever users leave the format string blank or unset with *Calends* methods.

### Unregister

There are two ways to unregister a calendar system once it's no longer needed. The first is to simply destruct the instance you created to register it. For that reason, it's important to store all your calendar systems to variables rather than simply constructing them in place. Well, that and the fact you need the calendar system object to persist in order to handle requests from the rest of the library.

The other way to unregister a calendar system is to do so manually, using the instance you created to register it in the first place:

```
Calends\CalendarDefinition::unregister()
```

Removes a calendar system from the callback list.

### Check and List

```
static Calends\CalendarDefinition::isRegistered(string $name) → bool
```

#### Parámetros

- **\$name** (string) – The calendar system name to check for.

**Devuelve** Whether or not the calendar system is currently registered.

**Tipo del valor devuelto** bool

Returns whether or not a calendar system has been registered, yet.

```
static Calends\CalendarDefinition::listRegistered → array
```

**Devuelve** The sorted list of calendar systems currently registered.

**Tipo del valor devuelto** [string]

Returns the list of calendar systems currently registered.

## 6.5.3 Types and Values

Now we get to the inner workings that make calendar systems function – even the built-in ones. The majority of the «magic» comes from the *TAITime* object itself, as a reliable way of storing the exact instants being calculated, and the only way times are handled by the library itself. A handful of methods provide basic operations that calendar system developers can use to simplify their conversions (adding and subtracting the values of other timestamps, and importing/exporting timestamp values from/to string and numeric types, in particular), and a couple of helpers exclusively handle adding and removing UTC leap second offsets. As long as you can convert your dates to/from Unix timestamps in a string or numeric type, the rest is handled entirely by these helpers in the library itself.

**class** Calends\TAITime

*TAITime* stores a TAI64NARUX instant in a reliable, easily-converted format. Each 9-digit fractional segment is stored in a separate 32-bit integer to preserve its value with a very high degree of accuracy, without having to rely on string parsing or external arbitrary-precision mathematics libraries.

**property** seconds(*float*)

The number of TAI seconds since CE 1970-01-01 00:00:00 TAI. Should be an integer value; the *float* type is used, here, only to be able to hold a full signed 64-bit integer value regardless of architecture.

---

**Nota:** TAI vs UTC

You may have noticed that a TAI64Time object stores times in TAI seconds, not Unix seconds, with a timezone offset of TAI rather than UTC. This distinction is **very important** as it will affect internal calculations and comparisons to mix the two up. TAI time is very similar to Unix time (itself based on UTC time), with one major difference. While Unix/UTC seconds include the insertion and removal of «leap seconds» to keep the solar zenith at local noon (which is useful for day-to-day living and planning), TAI seconds are a continuous count, unconcerned with dates whatsoever. Indeed, the only reason a date was given in the description above was to make it easier for human readers to know exactly when 0 TAI took place.

In other words, once you have a Unix timestamp of your instant calculated, be sure to convert it using *fromUTC* before returning the result to the rest of the library. And then, of course, you'll also need to convert instants from the library back using *toUTC* before generating outputs.

---

**property** nano(*integer*)

The first 9 digits of the timestamp's fractional component.

**property** atto(*integer*)

The 10th through 18th digits of the fractional component.

**property** ronto(*integer*)

The 19th through 27th digits of the fractional component.

**property** udedcto(*integer*)

The 28th through 36th digits of the fractional component.

**property** xindecto(*integer*)

The 37th through 45th digits of the fractional component.

**add**(TAITime \$z) → TAITime**Parámetros**

- **\$z** (*TAITime*) – The timestamp to add to the current one.

**Devuelve** The sum of the two timestamps.

**Tipo del valor devuelto** *TAITime*

Calculates the sum of two *TAITime* values.

**sub**(TAITime \$z) → TAITime**Parámetros**

- **\$z** (*TAITime*) – The timestamp to subtract from the current one.

**Devuelve** The difference of the two timestamps.

**Tipo del valor devuelto** *TAITime*

Calculates the difference of two *TAITime* values.

**toString()** → string

**Devuelve** The decimal string representation of the current timestamp.

**Tipo del valor devuelto** string

Returns the decimal string representation of the *TAITime* value.

---

**Nota:** *TAITime* also implements `__toString`, so you can use that instead of calling this function directly, if you prefer.

---

**fromString(string \$in)** → *TAITime*

**Parámetros**

- **\$in** (string) – The decimal string representation of a timestamp to calculate.

**Devuelve** The calculated timestamp.

**Tipo del valor devuelto** *TAITime*

Calculates a *TAITime* from its decimal string representation.

**toHex()** → string

**Devuelve** The hexadecimal string representation of the current timestamp.

**Tipo del valor devuelto** string

Returns the hexadecimal string representation of the *TAITime* value.

**fromHex(string \$in)** → *TAITime*

**Parámetros**

- **\$in** (string) – The hexadecimal string representation of a timestamp to calculate.

**Devuelve** The calculated timestamp.

**Tipo del valor devuelto** *TAITime*

Calculates a *TAITime* from its hexadecimal string representation.

**toNumber()** → float

**Devuelve** The numeric representation of the current timestamp.

**Tipo del valor devuelto** float

Returns the float representation of the *TAITime* value.

**fromNumber(numeric \$in)** → *TAITime*

**Parámetros**

- **\$in** (integer or float) – The arbitrary-precision floating point representation of a timestamp to calculate.

**Devuelve** The calculated timestamp.

**Tipo del valor devuelto** *TAITime*

Calculates a *TAITime* from its numeric (integer or float) representation.

**fromUTC()** → TAItime

**Devuelve** The calculated timestamp.

**Tipo del valor devuelto** *TAItime*

Removes the UTC leap second offset from a TAItime value.

**toUTC()** → TAItime

**Devuelve** The calculated timestamp.

**Tipo del valor devuelto** *TAItime*

Adds the UTC leap second offset to a TAItime value.

### 7.1 Contributions

Pull requests are always welcome on [GitHub](#)! That said, please be open to discussing the PR content, and possibly revising it if requested. Not all requests can be merged, and not all changes are desired.

Or, you can contribute some money, instead! Check out [my Patreon](#) for options, there. Other options will likely be added for one-time donations in the future.

#### 7.1.1 Security Reporting

Report all security-related issues to [dan \(dot\) hunsaker \(plus\) calends \(at\) gmail](#), and use PGP or GPG protections on your message (the account's key is `44806AB9`, or you can look it up by the email address). Security issues will be addressed internally before making any vulnerability announcements.

#### 7.1.2 Contributors

##### Code

[@danhunsaker](#)

## **Patrons**

- Dave McGrath
- M. Fredette

### C

`calends`, [20](#)

`calends/calendars`, [77](#)



**C**

Calends, [60](#)



## Símbolos

- (calends.\*Calends) UnmarshalJSON (*Golang function*), [29](#)
- (calends.\*Calends) UnmarshalText (*Golang function*), [29](#)
- (calends.Calends) Abuts (*Golang function*), [26](#)
- (calends.Calends) Add (*Golang function*), [23](#)
- (calends.Calends) AddFromEnd (*Golang function*), [23](#)
- (calends.Calends) Compare (*Golang function*), [26](#)
- (calends.Calends) Contains (*Golang function*), [26](#)
- (calends.Calends) Date (*Golang function*), [21](#)
- (calends.Calends) Difference (*Golang function*), [25](#)
- (calends.Calends) Duration (*Golang function*), [21](#)
- (calends.Calends) EndDate (*Golang function*), [21](#)
- (calends.Calends) EndsAfter (*Golang function*), [29](#)
- (calends.Calends) EndsBefore (*Golang function*), [28](#)
- (calends.Calends) EndsDuring (*Golang function*), [28](#)
- (calends.Calends) Gap (*Golang function*), [25](#)
- (calends.Calends) Intersect (*Golang function*), [25](#)
- (calends.Calends) IsAfter (*Golang function*), [28](#)
- (calends.Calends) IsBefore (*Golang function*), [27](#)
- (calends.Calends) IsDuring (*Golang function*), [28](#)
- (calends.Calends) IsLonger (*Golang function*), [27](#)
- (calends.Calends) IsSame (*Golang function*), [26](#)
- (calends.Calends) IsSameDuration (*Golang function*), [27](#)
- (calends.Calends) IsShorter (*Golang function*), [27](#)
- (calends.Calends) MarshalJSON (*Golang function*), [29](#)
- (calends.Calends) MarshalText (*Golang function*), [29](#)
- (calends.Calends) Merge (*Golang function*), [25](#)
- (calends.Calends) Next (*Golang function*), [24](#)
- (calends.Calends) Overlaps (*Golang function*), [26](#)
- (calends.Calends) Previous (*Golang function*), [24](#)
- (calends.Calends) SetDate (*Golang function*), [22](#)
- (calends.Calends) SetDuration (*Golang function*), [22](#)
- (calends.Calends) SetDurationFromEnd (*Golang function*), [22](#)
- (calends.Calends) SetEndDate (*Golang function*), [22](#)
- (calends.Calends) StartsAfter (*Golang function*), [28](#)
- (calends.Calends) StartsBefore (*Golang function*), [27](#)
- (calends.Calends) StartsDuring (*Golang function*), [28](#)
- (calends.Calends) String (*Golang function*), [29](#)
- (calends.Calends) Subtract (*Golang function*), [23](#)
- (calends.Calends) SubtractFromEnd (*Golang function*), [24](#)
- (calends/calendars.CalendarDefinition) FromInternal (*Golang function*), [78](#)
- (calends/calendars.CalendarDefinition) Offset (*Golang function*), [78](#)
- (calends/calendars.CalendarDefinition) ToInternal (*Golang function*), [77](#)
- (calends/calendars.TAI64NARUXTIME) Add (*Golang function*), [80](#)
- (calends/calendars.TAI64NARUXTIME) Float (*Golang function*), [81](#)
- (calends/calendars.TAI64NARUXTIME) HexString (*Golang function*), [81](#)
- (calends/calendars.TAI64NARUXTIME) MarshalBinary (*Golang function*), [81](#)
- (calends/calendars.TAI64NARUXTIME) MarshalText (*Golang function*), [81](#)
- (calends/calendars.TAI64NARUXTIME) String (*Golang function*), [81](#)
- (calends/calendars.TAI64NARUXTIME) Sub (*Golang function*), [80](#)
- (calends/calendars.TAI64NARUXTIME) UnmarshalBinary (*Golang function*), [81](#)
- (calends/calendars.TAI64NARUXTIME)

UnmarshalText (*Golang function*), [81](#)

## A

abuts

calends-(batch-mode)opción de línea de comando, [18](#)

calends-compareopción de línea de comando, [12](#)

abuts() (*Calends\Calends method*), [66](#)

add

calends-(batch-mode)opción de línea de comando, [16](#)

add() (*Calends\Calends method*), [63](#)

add() (*Calends\TAITime method*), [100](#)

add-from-end

calends-(batch-mode)opción de línea de comando, [16](#)

addFromEnd() (*Calends\Calends method*), [63](#)

after

calends-compareopción de línea de comando, [13](#)

atto (*Calends\TAITime property*), [100](#)

## B

before

calends-compareopción de línea de comando, [13](#)

## C

C/C++

custom calendars, [83](#)

installation, [7](#)

usage, [30](#)

CalendarDefinition (*clase en Calends*), [98](#)

CalendarDefinition() (*clase*), [90](#), [94](#)

CalendarDefinition.defaultFormat (*atributo de CalendarDefinition*), [94](#)

CalendarDefinition.name (*atributo de CalendarDefinition*), [94](#)

calendars, [69](#)

Gregorian Calendar, [71](#)

Julian Day Count, [71](#)

Stardate, [72](#)

TAI64 Time, [74](#)

UNIX Time, [74](#)

Calends (*clase en Calends*), [60](#)

calends (*módulo*), [51](#)

Calends (*namespace*), [60](#), [97](#)

calends (*package*), [20](#), [41](#)

Calends() (*clase*), [41](#), [51](#)

calends.Calends (*Golang type*), [20](#)

calends.Create (*Golang function*), [20](#)

calends/calendars (*package*), [77](#)

calends/calendars.CalendarDefinition (*Golang type*), [77](#)

calends/calendars.ErrInvalidFormat (*Golang type*), [83](#)

calends/calendars.ErrUnknownCalendar (*Golang function*), [83](#)

calends/calendars.ErrUnsupportedInput (*Golang type*), [83](#)

calends/calendars.ListRegistered (*Golang function*), [79](#)

calends/calendars.Registered (*Golang function*), [79](#)

calends/calendars.RegisterElements (*Golang function*), [78](#)

calends/calendars.RegisterObject (*Golang function*), [78](#)

calends/calendars.TAI64NARUXTime (*Golang type*), [80](#)

calends/calendars.TAI64NARUXTimeFromDecimalString (*Golang function*), [82](#)

calends/calendars.TAI64NARUXTimeFromFloat (*Golang function*), [82](#)

calends/calendars.TAI64NARUXTimeFromHexString (*Golang function*), [82](#)

calends/calendars.TAItoUTC (*Golang function*), [82](#)

calends/calendars.Unregister (*Golang function*), [79](#)

calends/calendars.UTCtoTAI (*Golang function*), [82](#)

Calends\_abuts (*C function*), [37](#)

Calends\_add\_double (*C function*), [33](#)

Calends\_add\_from\_end\_double (*C function*), [34](#)

Calends\_add\_from\_end\_long\_long (*C function*), [34](#)

Calends\_add\_from\_end\_string (*C function*), [34](#)

Calends\_add\_long\_long (*C function*), [33](#)

Calends\_add\_string (*C function*), [33](#)

Calends\_calendar\_from\_internal (*C function*), [84](#)

Calends\_calendar\_list\_registered (*C function*), [85](#)

Calends\_calendar\_offset\_double (*C function*), [84](#)

Calends\_calendar\_offset\_long\_long (*C function*), [84](#)

Calends\_calendar\_offset\_string (*C function*), [84](#)

Calends\_calendar\_offset\_tai (*C function*), [84](#)

Calends\_calendar\_registered (*C function*), [85](#)

Calends\_calendar\_to\_internal\_double (*C function*), [83](#)

Calends\_calendar\_to\_internal\_long\_long (*C function*), [83](#)

Calends\_calendar\_to\_internal\_string (*C function*), [83](#)

Calends\_calendar\_to\_internal\_tai (*C function*), [83](#)

Calends\_calendar\_unregister (*C function*), [85](#)

Calends\_compare (*C function*), [36](#)

---

Calends\_contains (C function), 37  
 Calends\_create\_double (C function), 30  
 Calends\_create\_double\_end\_period (C function), 30  
 Calends\_create\_double\_range (C function), 30  
 Calends\_create\_double\_start\_period (C function), 30  
 Calends\_create\_long\_long (C function), 30  
 Calends\_create\_long\_long\_end\_period (C function), 30  
 Calends\_create\_long\_long\_range (C function), 30  
 Calends\_create\_long\_long\_start\_period (C function), 30  
 Calends\_create\_string (C function), 30  
 Calends\_create\_string\_end\_period (C function), 30  
 Calends\_create\_string\_range (C function), 30  
 Calends\_create\_string\_start\_period (C function), 30  
 Calends\_date (C function), 31  
 Calends\_decode\_json (C function), 41  
 Calends\_decode\_text (C function), 40  
 Calends\_difference (C function), 36  
 Calends\_duration (C function), 31  
 Calends\_encode\_json (C function), 40  
 Calends\_encode\_text (C function), 40  
 Calends\_end\_date (C function), 31  
 Calends\_ends\_after (C function), 40  
 Calends\_ends\_before (C function), 38  
 Calends\_ends\_during (C function), 39  
 Calends\_gap (C function), 36  
 Calends\_intersect (C function), 35  
 Calends\_is\_after (C function), 39  
 Calends\_is\_before (C function), 38  
 Calends\_is\_during (C function), 39  
 Calends\_is\_longer (C function), 38  
 Calends\_is\_same (C function), 37  
 Calends\_is\_same\_duration (C function), 38  
 Calends\_is\_shorter (C function), 37  
 Calends\_merge (C function), 35  
 Calends\_next\_double (C function), 35  
 Calends\_next\_long\_long (C function), 35  
 Calends\_next\_string (C function), 35  
 Calends\_overlaps (C function), 37  
 Calends\_previous\_double (C function), 35  
 Calends\_previous\_long\_long (C function), 35  
 Calends\_previous\_string (C function), 35  
 Calends\_register\_panic\_handler (C function), 41  
 Calends\_release (C function), 33  
 Calends\_starts\_after (C function), 39  
 Calends\_starts\_before (C function), 38  
 Calends\_starts\_during (C function), 39  
 Calends\_string (C function), 40  
 Calends\_subtract\_double (C function), 34  
 Calends\_subtract\_from\_end\_double (C function), 34  
 Calends\_subtract\_from\_end\_long\_long (C function), 34  
 Calends\_subtract\_from\_end\_string (C function), 34  
 Calends\_subtract\_long\_long (C function), 34  
 Calends\_subtract\_string (C function), 34  
 Calends\_with\_date\_double (C function), 32  
 Calends\_with\_date\_long\_long (C function), 32  
 Calends\_with\_date\_string (C function), 32  
 Calends\_with\_duration\_double (C function), 32  
 Calends\_with\_duration\_from\_end\_double (C function), 33  
 Calends\_with\_duration\_from\_end\_long\_long (C function), 33  
 Calends\_with\_duration\_from\_end\_string (C function), 33  
 Calends\_with\_duration\_long\_long (C function), 32  
 Calends\_with\_duration\_string (C function), 32  
 Calends\_with\_end\_date\_double (C function), 32  
 Calends\_with\_end\_date\_long\_long (C function), 32  
 Calends\_with\_end\_date\_string (C function), 32  
 calends-(batch-mode)opción de línea de comando  
   abuts, 18  
   add, 16  
   add-from-end, 16  
   compare, 18  
   contains, 18  
   date, 15  
   difference, 17  
   end-date, 15  
   ends-after, 20  
   ends-before, 19  
   ends-during, 19  
   gap, 17  
   intersect, 17  
   is-after, 19  
   is-before, 19  
   is-during, 19  
   is-longer, 18  
   is-same, 18  
   is-same-duration, 18  
   is-shorter, 18  
   merge, 17  
   next, 16  
   overlaps, 18  
   parse, 14  
   parse-range, 14  
   previous, 16  
   set-date, 15  
   set-end-date, 15  
   starts-after, 19

- starts-before, 19
- starts-during, 19
- subtract, 16
- subtract-from-end, 16
- calends-compareopción de línea de comando
  - abuts, 12
  - after, 13
  - before, 13
  - contains, 12
  - during, 13
  - end-after, 14
  - end-before, 13
  - end-during, 13
  - longer, 13
  - overlaps, 12
  - same, 12
  - same-duration, 12
  - shorter, 12
  - start-after, 14
  - start-before, 13
  - start-during, 13
- CalendsError() (*clase*), 60
- CalendsException (*clase en Calends*), 69
- CalendsException() (*clase*), 51
- calendsopción de línea de comando
  - convert, 11
  - format, 12
  - offset, 12
  - parse, 11
- CLI
  - installation, 7
  - usage, 11
- compare
  - calends-(batch-mode)opción de línea de comando, 18
- compare() (*Calends\Calends method*), 66
- contains
  - calends-(batch-mode)opción de línea de comando, 18
  - calends-compareopción de línea de comando, 12
- contains() (*Calends\Calends method*), 66
- convert
  - calendsopción de línea de comando, 11
- create() (*Calends\Calends method*), 61
- custom calendars, 75
  - C/C++, 83
  - Dart, 89
  - Golang, 77
  - JS, 93
  - PHP, 97
  - WASM, 93

## D

- Dart
  - custom calendars, 89
  - installation, 8
  - usage, 41
- date
  - calends-(batch-mode)opción de línea de comando, 15
- date() (*Calends\Calends method*), 61
- difference
  - calends-(batch-mode)opción de línea de comando, 17
- difference() (*Calends\Calends method*), 65
- duration() (*Calends\Calends method*), 62
- during
  - calends-compareopción de línea de comando, 13

## E

- end-after
  - calends-compareopción de línea de comando, 14
- end-before
  - calends-compareopción de línea de comando, 13
- end-date
  - calends-(batch-mode)opción de línea de comando, 15
- end-during
  - calends-compareopción de línea de comando, 13
- endDate() (*Calends\Calends method*), 61
- ends-after
  - calends-(batch-mode)opción de línea de comando, 20
- ends-before
  - calends-(batch-mode)opción de línea de comando, 19
- ends-during
  - calends-(batch-mode)opción de línea de comando, 19
- endsAfter() (*Calends\Calends method*), 68
- endsBefore() (*Calends\Calends method*), 67
- endsDuring() (*Calends\Calends method*), 68

## F

- format
  - calendsopción de línea de comando, 12
- fromHex() (*Calends\TAITime method*), 101
- fromInternal() (*Calends\CalendarDefinition method*), 98
- fromNumber() (*Calends\TAITime method*), 101
- fromString() (*Calends\TAITime method*), 101

fromUTC() (*Calends\TAITime method*), [101](#)

## G

gap

calends-(batch-mode)opción de línea de comando, [17](#)

gap() (*Calends\Calends method*), [65](#)

Golang

custom calendars, [77](#)

installation, [7](#)

usage, [20](#)

Gregorian Calendar, [71](#)

## I

installation, [6](#)

C/C++, [7](#)

CLI, [7](#)

Dart, [8](#)

Golang, [7](#)

JS, [8](#)

PHP, [9](#)

WASM, [8](#)

intersect

calends-(batch-mode)opción de línea de comando, [17](#)

intersect() (*Calends\Calends method*), [65](#)

is-after

calends-(batch-mode)opción de línea de comando, [19](#)

is-before

calends-(batch-mode)opción de línea de comando, [19](#)

is-during

calends-(batch-mode)opción de línea de comando, [19](#)

is-longer

calends-(batch-mode)opción de línea de comando, [18](#)

is-same

calends-(batch-mode)opción de línea de comando, [18](#)

is-same-duration

calends-(batch-mode)opción de línea de comando, [18](#)

is-shorter

calends-(batch-mode)opción de línea de comando, [18](#)

isAfter() (*Calends\Calends method*), [68](#)

isBefore() (*Calends\Calends method*), [67](#)

isDuring() (*Calends\Calends method*), [68](#)

isLonger() (*Calends\Calends method*), [67](#)

isRegistered() (*Calends\CalendarDefinition method*), [99](#)

isSame() (*Calends\Calends method*), [66](#)

isSameDuration() (*Calends\Calends method*), [67](#)

isShorter() (*Calends\Calends method*), [67](#)

## J

JDC, *véase* Julian Day Count

JS

custom calendars, [93](#)

installation, [8](#)

usage, [51](#)

jsonUnserialize() (*Calends\Calends method*), [69](#)

Julian Day Count, [71](#)

## L

listRegistered() (*Calends\CalendarDefinition method*), [99](#)

longer

calends-compareopción de línea de comando, [13](#)

## M

merge

calends-(batch-mode)opción de línea de comando, [17](#)

merge() (*Calends\Calends method*), [65](#)

## N

nano (*Calends\TAITime property*), [100](#)

next

calends-(batch-mode)opción de línea de comando, [16](#)

next() (*Calends\Calends method*), [64](#)

## O

offset

calendsopción de línea de comando, [12](#)

offset() (*Calends\CalendarDefinition method*), [98](#)

overlaps

calends-(batch-mode)opción de línea de comando, [18](#)

calends-compareopción de línea de comando, [12](#)

overlaps() (*Calends\Calends method*), [66](#)

## P

parse

calends-(batch-mode)opción de línea de comando, [14](#)

calendsopción de línea de comando, [11](#)

parse-range

calends-(batch-mode)opción de línea de comando, [14](#)

PHP

custom calendars, [97](#)

installation, 9  
usage, 60  
previous  
    calends-(batch-mode)opción de línea de  
        comando, 16  
previous() (*Calends\Calends method*), 64

## R

ronto (*Calends\TAITime property*), 100

## S

same  
    calends-compareopción de línea de  
        comando, 12  
same-duration  
    calends-compareopción de línea de  
        comando, 12  
seconds (*Calends\TAITime property*), 100  
set-date  
    calends-(batch-mode)opción de línea de  
        comando, 15  
set-end-date  
    calends-(batch-mode)opción de línea de  
        comando, 15  
shorter  
    calends-compareopción de línea de  
        comando, 12  
Stardate, 72  
start-after  
    calends-compareopción de línea de  
        comando, 14  
start-before  
    calends-compareopción de línea de  
        comando, 13  
start-during  
    calends-compareopción de línea de  
        comando, 13  
starts-after  
    calends-(batch-mode)opción de línea de  
        comando, 19  
starts-before  
    calends-(batch-mode)opción de línea de  
        comando, 19  
starts-during  
    calends-(batch-mode)opción de línea de  
        comando, 19  
startsAfter() (*Calends\Calends method*), 68  
startsBefore() (*Calends\Calends method*), 67  
startsDuring() (*Calends\Calends method*), 68  
sub() (*Calends\TAITime method*), 100  
subtract  
    calends-(batch-mode)opción de línea de  
        comando, 16  
subtract() (*Calends\Calends method*), 63

subtract-from-end  
    calends-(batch-mode)opción de línea de  
        comando, 16  
subtractFromEnd() (*Calends\Calends method*), 64

## T

TAI64 Time, 74  
TAI64Time (*C type*), 86  
TAI64Time() (*clase*), 91, 95  
TAI64Time.Atto (*atributo de TAI64Time*), 92  
TAI64Time.atto (*atributo de TAI64Time*), 96  
TAI64Time.atto (*C member*), 86  
TAI64Time.Nano (*atributo de TAI64Time*), 92  
TAI64Time.nano (*atributo de TAI64Time*), 96  
TAI64Time.nano (*C member*), 86  
TAI64Time.padding (*C member*), 86  
TAI64Time.Ronto (*atributo de TAI64Time*), 92  
TAI64Time.ronto (*atributo de TAI64Time*), 96  
TAI64Time.ronto (*C member*), 86  
TAI64Time.Seconds (*atributo de TAI64Time*), 91  
TAI64Time.seconds (*atributo de TAI64Time*), 95  
TAI64Time.seconds (*C member*), 86  
TAI64Time.Udecto (*atributo de TAI64Time*), 92  
TAI64Time.udecto (*atributo de TAI64Time*), 96  
TAI64Time.udecto (*C member*), 86  
TAI64Time.Xindecto (*atributo de TAI64Time*), 92  
TAI64Time.xindecto (*atributo de TAI64Time*), 96  
TAI64Time.xindecto (*C member*), 86  
TAI64Time\_add (*C function*), 87  
TAI64Time\_decode\_binary (*C function*), 89  
TAI64Time\_decode\_text (*C function*), 88  
TAI64Time\_double (*C function*), 88  
TAI64Time\_encode\_binary (*C function*), 88  
TAI64Time\_encode\_text (*C function*), 88  
TAI64Time\_from\_double (*C function*), 88  
TAI64Time\_from\_hex\_string (*C function*), 88  
TAI64Time\_from\_string (*C function*), 87  
TAI64Time\_hex\_string (*C function*), 87  
TAI64Time\_string (*C function*), 87  
TAI64Time\_sub (*C function*), 87  
TAI64Time\_tai\_to\_utc (*C function*), 89  
TAI64Time\_utc\_to\_tai (*C function*), 89  
TAITime (*clase en Calends*), 99  
toHex() (*Calends\TAITime method*), 101  
toInternal() (*Calends\CalendarDefinition method*),  
    98  
toNumber() (*Calends\TAITime method*), 101  
toString() (*Calends\TAITime method*), 101  
toUTC() (*Calends\TAITime method*), 102

## U

udecto (*Calends\TAITime property*), 100  
UNIX Time, 74

`unregister()` (*Calends\CalendarDefinition* method),  
[99](#)

usage, [9](#)  
  C/C++, [30](#)  
  CLI, [11](#)  
  Dart, [41](#)  
  Golang, [20](#)  
  JS, [51](#)  
  PHP, [60](#)  
  WASM, [51](#)

## W

### WASM

  custom calendars, [93](#)  
  installation, [8](#)  
  usage, [51](#)  
`withDate()` (*Calends\Calends* method), [62](#)  
`withDuration()` (*Calends\Calends* method), [62](#)  
`withDurationFromEnd()` (*Calends\Calends* method),  
  [63](#)  
`withEndDate()` (*Calends\Calends* method), [62](#)

## X

`xindecto` (*Calends\TAITime* property), [100](#)