icalendar

Release 5.0.12

unknown

Apr 27, 2024

CONTENTS

The icalendar package is a RFC 5545 compatible parser/generator for iCalendar files.

Homepage
https://icalendar.readthedocs.io

Code
https://github.com/collective/icalendar

Mailing list
https://github.com/collective/icalendar/issues

Dependencies
python-dateutil and pytz.

Compatible with
Python 2.7 and 3.4+

License
BSD

CHAPTER

ONE

QUICK GUIDE

To **install** the package, run:

```
pip install icalendar
```

You can open an `.ics` file and see all the events:

```
>>> import icalendar
>>> path_to_ics_file = "src/icalendar/tests/calendars/example.ics"
>>> with open(path_to_ics_file) as f:
...     calendar = icalendar.Calendar.from_ical(f.read())
>>> for event in calendar.walk('VEVENT'):
...     print(event.get("SUMMARY"))
New Year's Day
Orthodox Christmas
International Women's Day
```

Using this package, you can also create calendars from scratch or edit existing ones.

CHAPTER

TWO

VERSIONS AND COMPATIBILITY

`icalendar` is a critical project used by many. It has been there for a long time and maintaining long-term compatibility with projects conflicts partially with providing and using the features that the latest Python versions bring.

Since we pour more effort into maintaining and developing icalendar, we split the project into two:

- Branch 4.x with maximum compatibility to Python versions `2.7` and `3.4+`, `PyPy2` and `PyPy3`.
- Branch master with the compatibility to Python versions `3.7+` and `PyPy3`.

We expect the `master` branch with versions `5+` receive the latest updates and features, and the `4.x` branch the subset of security and bug fixes only. We recommend migrating to later Python versions and also providing feedback if you depend on the `4.x` features.

2.1 Related projects

- icalevents. It is built on top of icalendar and allows you to query iCal files and get the events happening on specific dates. It manages recurrent events as well.
- recurring-ical-events. Library to query an `ICalendar` object for events happening at a certain date or within a certain time.
- x-wr-timezone. Library to make `ICalendar` objects and files using the non-standard `X-WR-TIMEZONE` compliant with the standard (RFC 5545).

2.2 Contents

2.2.1 About

Max M had often needed to parse and generate iCalendar files. Finally, he got tired of writing ad-hoc tools. This package is his attempt at making an iCalendar package for Python. The inspiration has come from the email package in the standard lib, which he thinks is pretty simple, yet efficient and powerful.

The icalendar package is an RFC 5545-compatible parser/generator for iCalendar files.

2.2.2 Installing iCalendar

To install the icalendar package, use:

```
pip install icalendar
```

If installation is successful, you will be able to import the iCalendar package, like this:

```
>>> import icalendar
```

Development Setup

To start contributing changes to icalendar, you can clone the project to your file system using Git. You can fork the project first and clone your fork, too.

```
git clone https://github.com/collective/icalendar.git
cd icalendar
```

Installing Python

You will need a version of Python installed to run the tests and execute the code. The latest version of Python 3 should work and will be enough to get you started. If you like to run the tests with different Python versions, the following setup process should work the same.

Install Tox

First, install tox..

```
pip install tox
```

From now on, tox will manage Python versions and test commands for you.

Running Tests

`tox` manages all test environments in all Python versions.

To run all tests in all environments, simply run `tox`

```
tox
```

You may not have all Python versions installed or you may want to run a specific one. Have a look at the documentation. This is how you can run `tox` with Python 3.9:

```
tox -e py39
```

Accessing a `tox` environment

If you like to enter a specific tox environment, you can do this:

```
source .tox/py39/bin/activate
```

Install `icalendar` Manually

The best way to test the package is to use `tox` as described above. If for some reason you cannot install `tox`, you can go ahead with the following section using your installed version of Python and `pip`.

If for example, you would like to use your local copy of icalendar in another Python environment, this section explains how to do it.

You can install the local copy of `icalendar` with `pip` like this:

```
cd icalendar
python -m pip install -e .
```

This installs the module and dependencies in your Python environment so that you can access local changes. If tox fails to install `icalendar` during its first run, you can activate the environment in the `.tox` folder and manually setup `icalendar` like this.

Try it out:

```
Python 3.9.5 (default, Nov 23 2021, 15:27:38)
Type "help", "copyright", "credits" or "license" for more information.
>>> import icalendar
>>> icalendar.__version__
'5.0.12'
```

Building the documentation locally

To build the documentation follow these steps:

```
$ source .tox/py39/bin/activate
$ pip install -r requirements_docs.txt
$ cd docs
$ make html
```

You can now open the output from `_build/html/index.html`. To build the presentation-version use `make presentation` instead of `make html`. You can open the presentation at `presentation/index.html`.

You can also use `tox` to build the documentation:

```
cd icalendar
tox -e docs
```

2.2.3 iCalendar package

This package is used for parsing and generating iCalendar files following the standard in RFC 2445.

It should be fully compliant, but it is possible to generate and parse invalid files if you really want to.

File structure

An iCalendar file is a text file (utf-8) with a special format. Basically it consists of content lines.

Each content line defines a property that has 3 parts (name, parameters, values). Parameters are optional.

A simple content line with only name and value could look like this:

```
BEGIN:VCALENDAR
```

A content line with parameters can look like this:

```
ATTENDEE;CN=Max Rasmussen;ROLE=REQ-PARTICIPANT:MAILTO:example@example.com
```

And the parts are:

```
Name:   ATTENDEE
Params: CN=Max Rasmussen;ROLE=REQ-PARTICIPANT
Value:  MAILTO:example@example.com
```

Long content lines are usually "folded" to less than 75 character, but the package takes care of that.

Overview

On a higher level iCalendar files consists of components. Components can have sub components.

The root component is the VCALENDAR:

```
BEGIN:VCALENDAR
... vcalendar properties ...
END:VCALENDAR
```

The most frequent subcomponent to a VCALENDAR is a VEVENT. They are nested like this:

```
BEGIN:VCALENDAR
... vcalendar properties ...
BEGIN:VEVENT
... vevent properties ...
END:VEVENT
  END:VCALENDAR
```

Inside the components there are properties with values. The values have special types. Like integer, text, datetime etc. these values are encoded in a special text format in an iCalendar file.

There are methods for converting to and from these encodings in the package.

These are the most important imports:

```
>>> from icalendar import Calendar, Event
```

Components

Components are like (Case Insensitive) dicts. So if you want to set a property you do it like this. The calendar is a component:

```
>>> cal = Calendar()
>>> cal['dtstart'] = '20050404T080000'
>>> cal['summary'] = 'Python meeting about calendaring'
>>> for k,v in cal.items():
...     k,v
('DTSTART', '20050404T080000')
('SUMMARY', 'Python meeting about calendaring')
```

NOTE: the recommended way to add components to the calendar is to use create the subcomponent and add it via Calendar.add! The example above adds a string, but not a vText component.

You can generate a string for a file with the to_ical() method:

```
>>> cal.to_ical()
b'BEGIN:VCALENDAR\r\nDTSTART:20050404T080000\r\nSUMMARY:Python meeting about calendaring\
↪r\nEND:VCALENDAR\r\n'
```

The rendered view is easier to read:

```
BEGIN:VCALENDAR
DTSTART:20050404T080000
SUMMARY:Python meeting about calendaring
END:VCALENDAR
```

So, let's define a function so we can easily display to_ical() output:

```
>>> def display(cal):
...     return cal.to_ical().decode("utf-8").replace('\r\n', '\n').strip()
```

You can set multiple properties like this:

```
>>> cal = Calendar()
>>> cal['attendee'] = ['MAILTO:maxm@mxm.dk','MAILTO:test@example.com']
>>> print(display(cal))
BEGIN:VCALENDAR
ATTENDEE:MAILTO:maxm@mxm.dk
ATTENDEE:MAILTO:test@example.com
END:VCALENDAR
```

If you don't want to care about whether a property value is a list or a single value, just use the add() method. It will automatically convert the property to a list of values if more than one value is added. Here is an example:

```
>>> cal = Calendar()
>>> cal.add('attendee', 'MAILTO:maxm@mxm.dk')
>>> cal.add('attendee', 'MAILTO:test@example.com')
>>> print(display(cal))
BEGIN:VCALENDAR
ATTENDEE:MAILTO:maxm@mxm.dk
ATTENDEE:MAILTO:test@example.com
END:VCALENDAR
```

Note: this version doesn't check for compliance, so you should look in the RFC 2445 spec for legal properties for each component, or look in the icalendar/calendar.py file, where it is at least defined for each component.

Subcomponents

Any component can have subcomponents. Eg. inside a calendar there can be events. They can be arbitrarily nested. First by making a new component:

```
>>> event = Event()
>>> event['uid'] = '42'
>>> event['dtstart'] = '20050404T080000'
```

And then appending it to a "parent":

```
>>> cal.add_component(event)
>>> print(display(cal))
BEGIN:VCALENDAR
ATTENDEE:MAILTO:maxm@mxm.dk
ATTENDEE:MAILTO:test@example.com
BEGIN:VEVENT
DTSTART:20050404T080000
UID:42
END:VEVENT
END:VCALENDAR
```

Subcomponents are appended to the subcomponents property on the component:

```
>>> cal.subcomponents
[VEVENT({'UID': '42', 'DTSTART': '20050404T080000'})]
```

Value types

Property values are utf-8 encoded strings.

This is impractical if you want to use the data for further computation. The datetime format for example looks like this: '20050404T080000'. But the package makes it simple to parse and generate iCalendar formatted strings.

Basically you can make the add() method do the thinking, or you can do it yourself.

To add a datetime value, you can use Pythons built in datetime types, and the set the encode parameter to true, and it will convert to the type defined in the spec:

```
>>> from datetime import datetime
>>> cal.add('dtstart', datetime(2005,4,4,8,0,0))
>>> cal['dtstart'].to_ical()
b'20050404T080000'
```

If that doesn't work satisfactorily for some reason, you can also do it manually.

In 'icalendar.prop', all the iCalendar data types are defined. Each type has a class that can parse and encode the type.

So if you want to do it manually:

```
>>> from icalendar import vDatetime
>>> now = datetime(2005,4,4,8,0,0)
>>> vDatetime(now).to_ical()
b'20050404T080000'
```

So the drill is to initialise the object with a python built in type, and then call the "to_ical()" method on the object. That will return an ical encoded string.

You can do it the other way around too. To parse an encoded string, just call the "from_ical()" method, and it will return an instance of the corresponding Python type:

```
>>> vDatetime.from_ical('20050404T080000')
datetime.datetime(2005, 4, 4, 8, 0)

>>> dt = vDatetime.from_ical('20050404T080000Z')
>>> repr(dt)[:62]
'datetime.datetime(2005, 4, 4, 8, 0, tzinfo=<UTC>)'
```

You can also choose to use the decoded() method, which will return a decoded value directly:

```
>>> cal = Calendar()
>>> cal.add('dtstart', datetime(2005,4,4,8,0,0))
>>> cal['dtstart'].to_ical()
b'20050404T080000'
>>> cal.decoded('dtstart')
datetime.datetime(2005, 4, 4, 8, 0)
```

Property parameters

Property parameters are automatically added, depending on the input value. For example, for date/time related properties, the value type and timezone identifier (if applicable) are automatically added here:

```
>>> import pytz
>>> event = Event()
>>> event.add('dtstart', datetime(2010, 10, 10, 10, 0, 0,
...                               tzinfo=pytz.timezone("Europe/Vienna")))

>>> lines = event.to_ical().splitlines()
>>> assert (
...     b"DTSTART;TZID=Europe/Vienna:20101010T100000"
...     in lines)
```

You can also add arbitrary property parameters by passing a parameters dictionary to the add method like so:

```
>>> event = Event()
>>> event.add('X-TEST-PROP', 'tryout.',
...           parameters={'prop1':'val1', 'prop2':'val2'})
>>> lines = event.to_ical().splitlines()
>>> assert b"X-TEST-PROP;PROP1=val1;PROP2=val2:tryout." in lines
```

Example

Here is an example generating a complete iCal calendar file with a single event that can be loaded into the Mozilla calendar.

Init the calendar:

```
>>> cal = Calendar()
>>> from datetime import datetime
>>> import pytz
```

Some properties are required to be compliant:

```
>>> cal.add('prodid', '-//My calendar product//mxm.dk//')
>>> cal.add('version', '2.0')
```

We need at least one subcomponent for a calendar to be compliant:

```
>>> event = Event()
>>> event.add('summary', 'Python meeting about calendaring')
>>> event.add('dtstart', datetime(2005,4,4,8,0,0,tzinfo=pytz.utc))
>>> event.add('dtend', datetime(2005,4,4,10,0,0,tzinfo=pytz.utc))
>>> event.add('dtstamp', datetime(2005,4,4,0,10,0,tzinfo=pytz.utc))
```

A property with parameters. Notice that they are an attribute on the value:

```
>>> from icalendar import vCalAddress, vText
>>> organizer = vCalAddress('MAILTO:noone@example.com')
```

Automatic encoding is not yet implemented for parameter values, so you must use the 'v*' types you can import from the icalendar package (they're defined in `icalendar.prop`):

```
>>> organizer.params['cn'] = vText('Max Rasmussen')
>>> organizer.params['role'] = vText('CHAIR')
>>> event['organizer'] = organizer
>>> event['location'] = vText('Odense, Denmark')

>>> event['uid'] = '20050115T101010/27346262376@mxm.dk'
>>> event.add('priority', 5)

>>> attendee = vCalAddress('MAILTO:maxm@example.com')
>>> attendee.params['cn'] = vText('Max Rasmussen')
>>> attendee.params['ROLE'] = vText('REQ-PARTICIPANT')
>>> event.add('attendee', attendee, encode=0)

>>> attendee = vCalAddress('MAILTO:the-dude@example.com')
>>> attendee.params['cn'] = vText('The Dude')
>>> attendee.params['ROLE'] = vText('REQ-PARTICIPANT')
>>> event.add('attendee', attendee, encode=0)
```

Add the event to the calendar:

```
>>> cal.add_component(event)
```

Write to disk:

```
>>> import tempfile, os
>>> directory = tempfile.mkdtemp()
>>> f = open(os.path.join(directory, 'example.ics'), 'wb')
>>> f.write(cal.to_ical())
522
>>> f.close()
```

Print out the calendar:

```
>>> print(cal.to_ical().decode("utf-8"))
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//My calendar product//mxm.dk//
BEGIN:VEVENT
SUMMARY:Python meeting about calendaring
DTSTART:20050404T080000Z
DTEND:20050404T100000Z
DTSTAMP:20050404T001000Z
UID:20050115T101010/27346262376@mxm.dk
ATTENDEE;CN="Max Rasmussen";ROLE=REQ-PARTICIPANT:MAILTO:maxm@example.com
ATTENDEE;CN="The Dude";ROLE=REQ-PARTICIPANT:MAILTO:the-dude@example.com
LOCATION:Odense\, Denmark
ORGANIZER;CN="Max Rasmussen";ROLE=CHAIR:MAILTO:noone@example.com
PRIORITY:5
END:VEVENT
END:VCALENDAR
```

2.2.4 More documentation

Have a look at the tests of this package to get more examples. All modules and classes docstrings, which document how they work.

2.2.5 API Reference

icalendar.cal

Calendar is a dictionary like Python object that can render itself as VCAL files according to rfc2445.

These are the defined components.

class icalendar.cal.**Alarm**(**args*, ***kwargs*)

class icalendar.cal.**Calendar**(**args*, ***kwargs*)

This is the base object for an iCalendar file.

class icalendar.cal.**Component**(**args*, ***kwargs*)

Component is the base object for calendar, Event and the other components defined in RFC 2445. Normally you will not use this class directly, but rather one of the subclasses.

add(*name*, *value*, *parameters=None*, *encode=1*)

Add a property.

Parameters

- **name** (*string*) – Name of the property.
- **value** (*Python native type or icalendar property type.*) – Value of the property. Either of a basic Python type of any of the icalendar's own property types.
- **parameters** (*Dictionary*) – Property parameter dictionary for the value. Only available, if encode is set to True.
- **encode** (*Boolean*) – True, if the value should be encoded to one of icalendar's own property types (Fallback is "vText") or False, if not.

Returns
None

add_component(*component*)

Add a subcomponent to this component.

content_line(*name*, *value*, *sorted=True*)

Returns property as content line.

content_lines(*sorted=True*)

Converts the Component and subcomponents into content lines.

decoded(*name*, *default=[]*)

Returns decoded value of property.

classmethod from_ical(*st*, *multiple=False*)

Populates the component recursively from a string.

get_inline(*name*, *decode=1*)

Returns a list of values (split on comma).

is_empty()

Returns True if Component has no items or subcomponents, else False.

property_items(*recursive=True*, *sorted=True*)

Returns properties in this component and subcomponents as: [(name, value), ...]

set_inline(*name*, *values*, *encode=1*)

Converts a list of values into comma separated string and sets value to that.

to_ical(*sorted=True*)

Parameters
sorted – Whether parameters and properties should be lexicographically sorted.

walk(*name=None*)

Recursively traverses component and subcomponents. Returns sequence of same. If name is passed, only components with name will be returned.

class icalendar.cal.**ComponentFactory**(**args*, ***kwargs*)

All components defined in rfc 2445 are registered in this factory class. To get a component you can use it like this.

class icalendar.cal.**Event**(**args*, ***kwargs*)

class icalendar.cal.**FreeBusy**(**args*, ***kwargs*)

class icalendar.cal.**Journal**(**args*, ***kwargs*)

class icalendar.cal.**Timezone**(**args*, ***kwargs*)

to_tz()

convert this VTIMEZONE component to a pytz.timezone object

class icalendar.cal.**TimezoneDaylight**(**args*, ***kwargs*)

class icalendar.cal.**TimezoneStandard**(**args*, ***kwargs*)

class icalendar.cal.**Todo**(**args*, ***kwargs*)

icalendar.prop

This module contains the parser/generators (or coders/encoders if you prefer) for the classes/datatypes that are used in iCalendar:

4.2 Defined property parameters are:

ALTREP, CN, CUTYPE, DELEGATED-FROM, DELEGATED-TO, DIR, ENCODING, FMTTYPE, FBTYPE, LANGUAGE, MEMBER, PARTSTAT, RANGE, RELATED, RELTYPE, ROLE, RSVP, SENT-BY, TZID, VALUE

4.3 Defined value data types are:

BINARY, BOOLEAN, CAL-ADDRESS, DATE, DATE-TIME, DURATION, FLOAT, INTEGER, PERIOD, RECUR, TEXT, TIME, URI, UTC-OFFSET

---

iCalendar properties have values. The values are strongly typed. This module defines these types, calling val.to_ical() on them will render them as defined in rfc2445.

If you pass any of these classes a Python primitive, you will have an object that can render itself as iCalendar formatted date.

Property Value Data Types start with a 'v'. they all have an to_ical() and from_ical() method. The to_ical() method generates a text string in the iCalendar format. The from_ical() method can parse this format and return a primitive Python datatype. So it should always be true that:

x == vDataType.from_ical(VDataType(x).to_ical())

These types are mainly used for parsing and file generation. But you can set them directly.

class icalendar.prop.**FixedOffset**(*offset*, *name*)

Fixed offset in minutes east from UTC.

dst(*dt*)

datetime -> DST offset as timedelta positive east of UTC.

tzname(*dt*)

datetime -> string name of time zone.

utcoffset(*dt*)

datetime -> timedelta showing offset from UTC, negative values indicating West of UTC

class icalendar.prop.**LocalTimezone**

Timezone of the machine where the code is running.

dst(*dt*)

datetime -> DST offset as timedelta positive east of UTC.

tzname(*dt*)

datetime -> string name of time zone.

utcoffset(*dt*)

datetime -> timedelta showing offset from UTC, negative values indicating West of UTC

class icalendar.prop.**TimeBase**

Make classes with a datetime/date comparable.

class icalendar.prop.**TypesFactory**(**args*, ***kwargs*)

All Value types defined in rfc 2445 are registered in this factory class.

The value and parameter names don't overlap. So one factory is enough for both kinds.

for_property(*name*)

Returns a the default type for a property or parameter

from_ical(*name*, *value*)

Decodes a named property or parameter value from an icalendar encoded string to a primitive python type.

to_ical(*name*, *value*)

Encodes a named value from a primitive python type to an icalendar encoded string.

class icalendar.prop.**vBinary**(*obj*)

Binary property values are base 64 encoded.

class icalendar.prop.**vBoolean**(**args*, ***kwargs*)

Returns specific string according to state.

class icalendar.prop.**vCalAddress**(*value*, *encoding='utf-8'*)

This just returns an unquoted string.

class icalendar.prop.**vDDDLists**(*dt_list*)

A list of vDDDTypes values.

class icalendar.prop.**vDDDTypes**(*dt*)

A combined Datetime, Date or Duration parser/generator. Their format cannot be confused, and often values can be of either types. So this is practical.

class icalendar.prop.**vDate**(*dt*)

Render and generates iCalendar date format.

class icalendar.prop.**vDatetime**(*dt*)

Render and generates icalendar datetime format.

vDatetime is timezone aware and uses the pytz library, an implementation of the Olson database in Python. When a vDatetime object is created from an ical string, you can pass a valid pytz timezone identifier. When a vDatetime object is created from a python datetime object, it uses the tzinfo component, if present. Otherwise an timezone-naive object is created. Be aware that there are certain limitations with timezone naive DATE-TIME components in the icalendar standard.

class icalendar.prop.**vDuration**(*td*)

Subclass of timedelta that renders itself in the iCalendar DURATION format.

property dt

The time delta for compatibility.

class icalendar.prop.**vFloat**(**args*, ***kwargs*)

Just a float.

class icalendar.prop.**vFrequency**(*value*, *encoding='utf-8'*)

A simple class that catches illegal values.

class icalendar.prop.**vGeo**(*geo*)

A special type that is only indirectly defined in the rfc.

class icalendar.prop.**vInline**(*value*, *encoding='utf-8'*)

This is an especially dumb class that just holds raw unparsed text and has parameters. Conversion of inline values are handled by the Component class, so no further processing is needed.

class icalendar.prop.**vInt**(**args*, ***kwargs*)

Just an int.

class icalendar.prop.**vPeriod**(*per*)

A precise period of time.

property dt

Make this cooperate with the other vDDDTypes.

class icalendar.prop.**vRecur**(**args*, ***kwargs*)

Recurrence definition.

class icalendar.prop.**vText**(*value*, *encoding='utf-8'*)

Simple text.

class icalendar.prop.**vTime**(**args*)

Render and generates iCalendar time format.

class icalendar.prop.**vUTCOffset**(*td*)

Renders itself as a utc offset.

class icalendar.prop.**vUri**(*value*, *encoding='utf-8'*)

Uniform resource identifier is basically just an unquoted string.

class icalendar.prop.**vWeekday**(*value*, *encoding='utf-8'*)

This returns an unquoted weekday abbrevation.

2.2.6 iCalendar utility

To get more information about the command line interface, use the `-h` option:

```
$ icalendar -h
```

view

To output a human readable summary of an event:

```
$ icalendar myfile.ics
```

The output will look something like this:

```
Organiser: Secretary <secretary@company.com>
Attendees:
  John Doe <j.doe@company.com>
```

(continues on next page)

(continued from previous page)

```
  Randy <boss@company.com>
Summary: Yearly evaluation.
When: Tue 14 Mar 2017 11:00-12:00
Location: Randy's office
Comment: Reminder.
Description:

Your yearly evaluation is scheduled for next Tuesday. Please be on time.
```

To use this in terminal based e-mail clients like mutt, add a new mime type (as root):

```
# cat << EOF > /usr/lib/mime/packages/icalendar
text/calendar; icalendar '%s'; copiousoutput; description=iCalendar text; priority=2
EOF
# update-mime
```

2.2.7 icalendar contributors

- Johannes Raggam <johannes@raggam.co.at> (Maintainer)
- Max M <maxm@mxm.dk> (Original author)
- Andreas Zeidler <az@zitc.de>
- Andrey Nikolaev <nikolaeff@gmail.com>
- Barak Michener <me@barakmich.com>
- Christian Geier <contact@lostpackets.de>
- Christophe de Vienne <cdevienne@gmail.com>
- Dai MIKURUBE <dmikurube@acm.org>
- Dan Stovall <dbstovall@gmail.com>
- Eric Hanchrow <erich@cozi.com>
- Eric Wieser <wieser.eric@gmail.com>
- Erik Simmler <tgecho@gmail.com>
- George V. Reilly <george@reilly.org>
- Jannis Leidel <jannis@leidel.info>
- Jeroen F.J. Laros <jlaros@fixedpoint.nl>
- Jeroen van Meeuwen (Kolab Systems) <vanmeeuwen@kolabsys.com>
- Jochen Sprickerhof <icalendar@jochen.sprickerhof.de>
- Jordan Kiang <jordan@cozi.com>
- Klaus Klein <kleink+github@kleink.org>
- Laurent Lasudry <lasudry@gmail.com>
- Lennart Regebro <lregebro@nuxeo.com>
- Léo S <leo@naeka.fr>
- Marc Egli <frog32@me.com>

- Markus Unterwaditzer <markus@unterwaditzer.net>
- Martijn Faassen <faassen@infrae.com>
- Martin Melin <git@martinmelin.com>
- Michael Smith <msmith@fluendo.com>
- Mikael Frykholm <mikael@frykholm.com>
- Olivier Grisel <ogrisel@nuxeo.com>
- Pavel Repin <prepin@gmail.com>
- Pedro Ferreira <jose.pedro.ferreira@cern.ch>
- Rembane <andeke@gmail.com>
- Robert Niederreiter <rnix@squarewave.at>
- Rok Garbas <rok@garbas.si>
- Ronan Dunklau <ronan@dunklau.fr>
- Russ <russ@rw.id.au>
- Sidnei da Silva <sidnei@enfoldsystems.com>
- Stanislav Láznička <slaznick@redhat.com>
- Stanislav Ochotnicky <sochotnicky@redhat.com>
- Stefan Schwarzer <sschwarzer@sschwarzer.net>
- Thomas Bruederli <thomas@roundcube.net>
- Thomas Weißschuh <thomas@t-8ch.de>
- Victor Varvaryuk <victor.varvariuc@gmail.com>
- Ville Skyttä <ville.skytta@iki.fi>
- Wichert Akkerman <wichert@wiggy.net>
- cillianderoiste <cillian.deroiste@gmail.com>
- fitnr <fitnr@fakeisthenewreal>
- hajdbo <boris@hajduk.org>
- ilya <ilya@boltnev-pc.(none)>
- spanktar <spanky@kapanka.com>
- tgecho <tgecho@gmail.com>
- tisto <tisto@plone.org>
- TomTry <tom.try@gmail.com>
- Andreas Ruppen <andreas.ruppen@gmail.com>
- Clive Stevens <clivest2@gmail.com>
- Dalton Durst <github@daltondur.st>
- Kamil Mańkowski <kam193@wp.pl>
- Tobias Brox <tobias@redpill-linpro.com>
- Nicco Kunzmann

- Robert Spralja <robert.spralja@gmail.com>
- Maurits van Rees <maurits@vanrees.org>
- jacadzaca <vitouejj@gmail.com>
- Mauro Amico <mauro.amico@gmail.com>
- Alexander Pitkin <peleccom@gmail.com>
- Michał Górny <mgorny@gentoo.org>
- Pronoy <lukex9442@gmail.com>
- Abe Hanoka <abe@habet.dev>
- **`Natasha Mattson <https://github.com/natashamm`_**
- NikEasY
- Matt Lewis <git@semiprime.com>
- Felix Stupp <felix.stupp@banananet.work>

Find out who contributed:

```
$ git shortlog -s -e
```

2.2.8 Maintenance

The goal of this section is to make sure that the `icalendar` library receives a clear maintenance structure with it that is transparent.

Maintainers

Currently, the maintainers are

- @geier
- @jacadzaca
- @niccokunzmann

Maintainers need this:

- **`Admin` access to the repository.**
  These can be enabled by a current maintainer or an GitHub organisation administrator in the settings.
- **`Maintainer` or `Owner` access to the PyPI project.**
  The new maintainer needs a PyPI account for this with Two Factor Authentication (2FA) enabled because `icalendar` is a critical project on PyPI. The access can be given in the collaboration Section on PyPI.
- **`Maintainer` access to the Read The Docs project.**
  This can be given by existing maintainers listed on the project's page. TODO: link to the settings
- **`PyPI environment access for GitHub Actions` grant new releases from tags.**
  This access can be granted in Settings → Environments → PyPI by adding the GitHub username to the list of "Required Reviewers".

Collaborators

Collaborators are people with write access to the repository. As a collaborator, you can

- merge Pull Requests,
- initiate a new release, see below.

We want to have as many knowledgeable people with write access taking responsibility as possible for these reasons:

- a constant flow of fixes and new features
- better code review
- lower bus factor and backup
- future maintainers

Nobody should merge their own pull requests. If you like to review or merge pull requests of other people and you have shown that you know how to contribute, you can ask for becoming a collaborator or a maintainer asks you if you would like to become one.

New Releases

This explains how to create a new release on PyPI.

Since collaborators and maintainers have write access the the repository, they can start the release process. However, only people with `PyPI environment access for GitHub Actions` can approve an automated release to PyPI.

1. Check that the `CHANGES.rst` is up to date with the latest merged pull requests and the version you want to release is correctly named.
2. Change the `__version__` variable in
   - the `src/icalendar/__init__.py` file and
   - in the `docs/install.rst` file (look for `icalendar.__version__`)
3. Create a commit on the `release` branch (or equivalent) to release this version.

   ```
   git checkout master
   git pull
   git checkout -b release master
   git add CHANGES.rst src/icalendar/__init__.py docs/install.rst
   git commit -m"version 5.0.0"
   ```

4. Push the commit and create a pull request Here is an example pull request #457.

   ```
   git push -u origin release
   ```

5. See if the CI-tests are running on the pull request. If they are not running, no new release can be issued. If the tests are running, merge the pull request.
6. Clean up behind you!

   ```
   git checkout master
   git pull
   git branch -d release
   git push -d origin release
   ```

7. Create a tag for the release and see if the CI-tests are running.

```
git checkout master
git pull
git tag v5.0.0
git push upstream v5.0.0 # could be origin or whatever reference
```

8. Once the tag is pushed and its CI-tests are passing, maintainers will get an e-mail:

```
Subject: Deployment review in collective/icalendar

tests: PyPI is waiting for your review
```

9. If the release is approved by a maintainer. It will be pushed to PyPI. If that happens, notify the issues that were fixed about this release.

10. Copy this to the start of `CHANGES.rst`:

```
5.0.2 (unreleased)
------------------

Minor changes:

- ...

Breaking changes:

- ...

New features:

- ...

Bug fixes:

- ...
```

11. Push the new CHANGELOG so it is used for future changes.

```
git checkout master
git pull
git add CHANGES.rst
git commit -m"Add new CHANGELOG section for future release

See https://icalendar.readthedocs.io/en/latest/maintenance.html#new-releases"
git push upstream master # could be origin or whatever reference
```

Links

This section contains useful links for maintainers and collaborators:

- Future of icalendar, looking for maintainer #360
- Comment on the Plone tests running with icalendar

2.2.9 License

2.2.10 Changelog

5.0.13 (unreleased)

Minor changes:

- Add funding information
- Update windows to olson conversion for Greenland Standard Time

Breaking changes:

- …

New features:

- Create GitHub releases for each tag.

Bug fixes:

- …

5.0.12 (2024-03-19)

Minor changes:

- Analyse code coverage of test files
- Added corpus to fuzzing directory
- Added exclusion of fuzzing corpus in MANIFEST.in
- Augmented fuzzer to optionally convert multiple calendars from a source string
- Add script to convert OSS FUZZ test cases to Python/pytest test cases
- Added additional exception handling of defined errors to fuzzer, to allow fuzzer to explore deeper
- Added more instrumentation to fuzz-harness
- Rename “contributor” to “collaborator” in documentation
- Correct the outdated “icalendar view myfile.ics” command in documentation. #588
- Update GitHub Actions steps versions
- Keep GitHub Actions up to date with GitHub’s Dependabot

Breaking changes:

- …

New features:

- …

Bug fixes:

- …
- Fixed index error in cal.py when attempting to pop from an empty stack
- Fixed type error in prop.py when attempting to join strings into a byte-string
- Caught Wrong Date Format in ical_fuzzer to resolve fuzzing coverage blocker

5.0.11 (2023-11-03)

Minor changes:

- The cli utility now displays start and end datetimes in the user’s local timezone. Ref: #561 [vimpostor]

New features:

- Added fuzzing harnesses, for integration to OSSFuzz.
- icalendar releases are deployed to Github releases Fixes: #563 [jacadzaca]

Bug fixes:

- CATEGORIES field now accepts a string as argument Ref: #322 [jacadzaca]
- Multivalue FREEBUSY property is now parsed properly Ref: #27 [jacadzaca]
- Compare equality and inequality of calendars more completely Ref: #570
- Use non legacy timezone name. Ref: #567
- Add some compare functions. Ref: #568
- Change OSS Fuzz build script to point to harnesses in fuzzing directory Ref: #574

5.0.10 (2023-09-26)

Bug fixes:

- Component._encode stops ignoring parameters argument on native values, now merges them Fixes: #557 [zocker1999net]

5.0.9 (2023-09-24)

Bug fixes:

- PERIOD values now set the timezone of their start and end. #556

5.0.8 (2023-09-18)

Minor changes:

- Update build configuration to build readthedocs. #538
- No longer run the `plone.app.event` tests.
- Add documentation on how to parse `.ics` files. #152
- Move pip caching into Python setup action.
- Check that issue #165 can be closed.
- Updated about.rst for issue #527
- Avoid `vText.__repr__` BytesWarning.

Bug fixes:

- Calendar components are now properly compared Ref: #550 Fixes: #526 [jacadzaca]

5.0.7 (2023-05-29)

Bug fixes:

- to_ical() now accepts RRULE BYDAY values>=10 #518

5.0.6 (2023-05-26)

Minor changes:

- Adjusted duration regex

5.0.5 (2023-04-13)

Minor changes:

- Added support for BYWEEKDAY in vRecur ref: #268

Bug fixes:

- Fix problem with ORGANIZER in FREE/BUSY #348

5.0.4 (2022-12-29)

Minor changes:

- Improved documentation Ref: #503, #504

Bug fixes:

- vBoolean can now be used as an parameter Ref: #501 Fixes: #500 [jacadzaca]

5.0.3 (2022-11-23)

New features:

- vDDDTypes is hashable #487 #492 [niccokunzmann]

Bug fixes:

- vDDDTypes' equality also checks the dt attribute #497 #492 [niccokunzmann]

5.0.2 (2022-11-03)

Minor changes:

- Refactored cal.py, tools.py and completed remaining minimal refactoring in parser.py. Ref: #481 [pronoym99]
- Calendar.from_ical no longer throws long errors Ref: #473 Fixes: #472 [jacadzaca]
- Make datetime value shorter by removing the value parameter where possible. Fixes: #318 [jacadzaca], [niccokunzmann]

New features:

- source code in documentation is tested using doctest #445 [niccokunzmann]

Bug fixes:

- broken properties are not added to the parent component Ref: #471 Fixes: #464 [jacadzaca]

5.0.1 (2022-10-22)

Minor changes:

- fixed setuptools deprecation warnings [mgorny]

Bug fixes:

- a well-known timezone timezone prefixed with a / is treated as if the slash wasn't present Ref: #467 Fixes: #466 [jacadzaca]

5.0.0 (2022-10-17)

Minor changes:

- removed deprecated test checks [tuergeist]
- Fix: cli does not support DURATION #354 [mamico]
- Add changelog and contributing to readthedocs documentation #428 [peleccom]
- fixed small typos #323 [rohnsha0]
- unittest to parametrized pytest refactoring [jacadzaca]

Breaking changes:

- Require Python 3.7 as minimum Python version. [maurits] [niccokunzmann]
- icalendar now takes a ics file directly as an input
- icalendar's CLI utility program's output is different
- Drop Support for Python 3.6. Versions 3.7 - 3.11 are supported and tested.

New features:

- icalendar utility outputs a 'Duration' row
- icalendar can take multiple ics files as an input

Bug fixes:

- Changed tools.UIDGenerator instance methods to static methods Ref: #345 [spralja]
- proper handling of datetime objects with *tzinfo* generated through zoneinfo.ZoneInfo. Ref: #334 Fixes: #333 [tobixen]
- Timestamps in UTC does not need tzid Ref: #338 Fixes: #335 [tobixen]
- add `__eq__` to `icalendar.prop.vDDDTypes` #391 [jacadzaca]
- Refactor deprecated unittest aliases for Python 3.11 compatibility #330 [tirkarthi]

5.0.0a1 (2022-07-11)

Breaking changes:

- Drop support for Python 3.4, 3.5 and PyPy2. [maurits]

New features:

- Document development setup Ref: #358 [niccokunzmann]

Bug fixes:

- Test with GitHub Actions. [maurits]

4.1.0 (2022-07-11)

New features:

- No longer test on Python 3.4, 3.5 and PyPy2, because we cannot get it to work. Technically it should still work, it is just no longer tested. Do not expect much development on branch 4.x anymore. The master branch will be for the remaining Python versions that we support. [maurits]

Bug fixes:

- Test with GitHub Actions. [maurits]

4.0.9 (2021-10-16)

Bug fixes:

- Fix vCategories for correct en/de coding. [thet]
- vDuration property value: Fix changing duration sign after multiple `to_ical` calls. Ref: #320 Fixes: #319 [barlik]

4.0.8 (2021-10-07)

Bug fixes:

- Support added for Python 3.9 and 3.10 (no code changes needed).
- Replace bare 'except:' with 'except Exception:' (#281)

4.0.7 (2020-09-07)

Bug fixes:

- fixed rrule handling, re-enabled test_create_america_new_york()

4.0.6 (2020-05-06)

Bug fixes:

- Use `vText` as default type, when convert recurrence definition to ical string. [kam193]

4.0.5 (2020-03-21)

Bug fixes:

- Fixed a docs issue related to building on Read the Docs [davidfischer]

4.0.4 (2019-11-25)

Bug fixes:

- Reduce Hypothesis iterations to speed up testing, allowing PRs to pass [UniversalSuperBox]

4.0.3 (2018-10-10)

Bug fixes:

- Categories are comma separated not 1 per line #265. [cleder]
- mark test with mixed timezoneaware and naive datetimes as an expected failure. [cleder]

4.0.2 (2018-06-20)

Bug fixes:

- Update all pypi.python.org URLs to pypi.org [jon.dufresne]

4.0.1 (2018-02-11)

- Added rudimentary command line interface. [jfjlaros]
- Readme, setup and travis updates. [jdufresne, PabloCastellano]

4.0.0 (2017-11-08)

Breaking changes:

- Drop support for Python 2.6 and 3.3.

3.12 (2017-11-07)

New features:

- Accept Windows timezone identifiers as valid. #242 [geier]

Bug fixes:

- Fix ResourceWarnings in setup.py when Python warnings are enabled. #244 [jdufresne]
- Fix invalid escape sequences in string and bytes literals. #245 [jdufresne]
- Include license file in the generated wheel package. #243 [jdufresne]
- Fix non-ASCII TZID and TZNAME parameter handling. #238 [clivest]
- Docs: update install instructions. #240 [Ekran]

3.11.7 (2017-08-27)

New features:

- added vUTCOffset.ignore_exceptions to allow surpressing of failed TZOFFSET parsing (for now this ignores the check for offsets > 24h) [geier]

3.11.6 (2017-08-04)

Bug fixes:

- Fix VTIMEZONEs including RDATEs #234. [geier]

3.11.5 (2017-07-03)

Bug fixes:

- added an assertion that VTIMEZONE sub-components' DTSTART must be of type DATETIME [geier]
- Fix handling of VTIMEZONEs with subcomponents with the same DTSTARTs and OFFSETs but which are of different types [geier]

3.11.4 (2017-05-10)

Bug fixes:

- Don't break on parameter values which contain equal signs, e.g. base64 encoded binary data [geier]
- Fix handling of VTIMEZONEs with subcomponents with the same DTSTARTs. [geier]

3.11.3 (2017-02-15)

Bug fixes:

- Removed `setuptools` as a dependency as it was only required by setup.py and not by the package.
- Don't split content lines on the unicode `LINE SEPARATOR` character `\u2028` but only on `CRLF` or `LF`.

3.11.2 (2017-01-12)

Bug fixes:

- Run tests with python 3.5 and 3.6. [geier]
- Allow tests failing with pypy3 on travis.ci. [geier]

3.11.1 (2016-12-19)

Bug fixes:

- Encode error message before adding it to the stack of collected error messages.

3.11 (2016-11-18)

Fixes:

- Successfully test with pypy and pypy3. [gforcada]
- Minor documentation update. [tpltnt]

3.10 (2016-05-26)

New:

- Updated components description to better comply with RFC 5545. Refs #183. [stlaz]
- Added PERIOD value type to date types. Also fixes incompatibilities described in #184. Refs #189. [stlaz]

Fixes:

- Fix testsuite for use with `dateutil>=2.5`. Refs #195. [untitaker]
- Reintroduce cal.Component.is_broken that was removed with 3.9.2. Refs #185. [geier]

3.9.2 (2016-02-05)

New:

- Defined `test_suite` in setup.py. Now tests can be run via `python setup.py test`. [geier]

Fixes:

- Fixed cal.Component.from_ical() representing an unknown component as one of the known. [stlaz]
- Fixed possible IndexError exception during parsing of an ical string. [stlaz]
- When doing a boolean test on `icalendar.cal.Component`, always return `True`. Before it was returning `False` due to CaselessDict, if it didn't contain any items. [stlaz]
- Fixed date-time being recognized as date or time during parsing. Added better error handling to parsing from ical strings. [stlaz]
- Added __version__ attribute to init.py. [TomTry]
- Documentation fixes. [TomTry]
- Pep 8, UTF 8 headers, dict/list calls to literals. [thet]

3.9.1 (2015-09-08)

- Fix `vPeriod.__repr__`. [spacekpe]
- Improve foldline() performance. This improves the foldline performance, especially for large strings like base64-encoded inline attachements. In some cases (1MB string) from 7 Minutes to less than 20ms for ASCII data and 500ms for non-ASCII data. Ref: #163. [emfree]

3.9.0 (2015-03-24)

- Creating timezone objects from VTIMEZONE components. [geier]
- Make `python-dateutil` a dependency. [geier]
- Made RRULE tolerant of trailing semicolons. [sleeper]
- Documentation fixes. [t-8ch, thet]

3.8.4 (2014-11-01)

- Add missing BYWEEKNO to recurrence rules. [russkel]

3.8.3 (2014-08-26)

- PERCENT property in VTODO renamed to PERCENT-COMPLETE, according to RFC5545. [thomascube]

3.8.2 (2014-07-22)

- Exclude editor backup files from egg distributions. Fixes #144. [thet]

3.8.1 (2014-07-17)

- The representation of CaselessDicts in 3.8 changed the name attribute of Components and therefore broke the external API. This has been fixed. [untitaker]

3.8 (2014-07-17)

- Allow dots in property names (Needed for vCard compatibility). Refs #143. [untitaker]
- Change class representation for CaselessDict objects to always include the class name or the class' name attribute, if available. Also show subcomponents for Component objects. [thet]
- Don't use data_encode for CaselessDict class representation but use dict's __repr__ method. [t-8ch]
- Handle parameters with multiple values, which is needed for vCard 3.0. Refs #142. [t-8ch]

3.7 (2014-06-02)

- For components with `ignore_exceptions` set to `True`, mark unparseable lines as broken instead rising a `ValueError`. `VEVENT` components have `ignore_exceptions` set to `True` by default. Ref #131. Fixes #104. [jkiang13]
- Make `python-dateutil` a soft-dependency. [boltnev]
- Add optional `sorted` parameter to `Component.to_ical`. Setting it to false allows the user to preserve the original property and parameter order. Ref #136. Fixes #133. [untitaker]
- Fix tests for latest `pytz`. Don't set `tzinfo` directly on datetime objects, but use pytz's `localize` function. Ref #138. [untitaker, thet]
- Remove incorrect use of __all__. We don't encourage using `from package import *` imports. Fixes #129. [eric-wieser]

3.6.2 (2014-04-05)

- Pep8 and cleanup. [lasudry]

3.6.1 (2014-01-13)

- Open text files referenced by setup.py as utf-8, no matter what the locale settings are set to. Fixes #122. [sochotnicky]
- Add tox.ini to source tarball, which simplifies testing for in distributions. [sochotnicky]

3.6 (2014-01-06)

- Python3 (3.3+) + Python 2 (2.6+) support [geier]
- Made sure to_ical() always returns bytes [geier]
- Support adding lists to a component property, which value already was a list and remove the Component.set method, which was only used by the add method. [thet]
- Remove ability to add property parameters via a value's params attribute when adding via cal.add (that was only possible for custom value objects and makes up a strange API), but support a parameter attribute on cal.add's method signature to pass a dictionary with property parameter key/value pairs. Fixes #116. [thet]
- Backport some of Regebro's changes from his regebro-refactor branch. [thet]
- Raise explicit error on another malformed content line case. [hajdbo]
- Correctly parse datetime component property values with timezone information when parsed from ical strings. [untitaker]

3.5 (2013-07-03)

- Let to_unicode be more graceful for non-unicode strings, as like CMFPlone's safe_unicode does it. [thet]

3.4 (2013-04-24)

- Switch to unicode internally. This should fix all en/decoding errors. [thet]
- Support for non-ascii parameter values. Fixes #88. [warvariuc]
- Added functions to transform chars in string with '\' + any of r',;:' chars into '%{:02X}' form to avoid splitting on chars escaped with '\'. [warvariuc]
- Allow seconds in vUTCOffset properties. Fixes #55. [thet]
- Let `Component.decode` better handle vRecur and vDDDLists properties. Fixes #70. [thet]
- Don't let `Component.add` re-encode already encoded values. This simplifies the API, since there is no need explicitly pass `encode=False`. Fixes #82. [thet]
- Rename tzinfo_from_dt to tzid_from_dt, which is what it does. [thet]
- More support for dateutil parsed tzinfo objects. Fixes #89. [leo-naeka]
- Remove python-dateutil version fix at all. Current python-dateutil has Py3 and Py2 compatibility. [thet]
- Declare the required python-dateutil dependency in setup.py. Fixes #90. [kleink]
- Raise test coverage. [thet]
- Remove interfaces module, as it is unused. [thet]
- Remove `test_doctests.py`, test suite already created properly in `test_icalendar.py`. [rnix]
- Transformed doctests into unittests, Test fixes and cleanup. [warvariuc]

3.3 (2013-02-08)

- Drop support for Python < 2.6. [thet]
- Allow vGeo to be instantiated with list and not only tuples of geo coordinates. Fixes #83. [thet]
- Don't force to pass a list to vDDDLists and allow setting individual RDATE and EXDATE values without having to wrap them in a list. [thet]
- Fix encoding function to allow setting RDATE and EXDATE values and not to have bypass encoding with an icalendar property. [thet]
- Allow setting of timezone for vDDDLists and support timezone properties for RDATE and EXDATE component properties. [thet]
- Move setting of TZID properties to vDDDTypes, where it belongs to. [thet]
- Use @staticmethod decorator instead of wrapper function. [warvariuc, thet]
- Extend quoting of parameter values to all of those characters: '",;: â'". This fixes an outlook incompatibility with some characters. Fixes: #79, Fixes: #81. [warvariuc]
- Define VTIMETZONE subcomponents STANDARD and DAYLIGHT for RFC5545 compliance. [thet]

3.2 (2012-11-27)

- Documentation file layout restructuring. [thet]
- Fix time support. vTime events can be instantiated with a datetime.time object, and do not inherit from datetime.time itself. [rdunklau]
- Correctly handle tzinfo objects parsed with dateutil. Fixes #77. [warvariuc, thet]
- Text values are escaped correclty. Fixes #74. [warvariuc]
- Returned old folding algorithm, as the current implementation fails in some cases. Fixes #72, Fixes #73. [warvariuc]
- Supports to_ical() on date/time properties for dates prior to 1900. [cdevienne]

3.1 (2012-09-05)

- Make sure parameters to certain properties propagate to the ical output. [kanarip]
- Re-include doctests. [rnix]
- Ensure correct datatype at instance creation time in `prop.vCalAddress` and `prop.vText`. [rnix]
- Apply TZID parameter to datetimes parsed from RECURRENCE-ID [dbstovall]
- Localize datetimes for timezones to avoid DST transition errors. [dbstovall]
- Allow UTC-OFFSET property value data types in seconds, which follows RFC5545 specification. [nikolaeff]
- Remove utctz and normalized_timezone methods to simplify the codebase. The methods were too tiny to be useful and just used at one place. [thet]
- When using Component.add() to add icalendar properties, force a value conversion to UTC for CREATED, DTSTART and LAST-MODIFIED. The RFC expects UTC for those properties. [thet]
- Removed last occurrences of old API (from_string). [Rembane]
- Add 'recursive' argument to property_items() to switch recursive listing. For example when parsing a text/calendar text including multiple components (e.g. a VCALENDAR with 5 VEVENTs), the previous situation required us to look over all properties in VEVENTs even if we just want the properties under the VCALENDAR component (VERSION, PRODID, CALSCALE, METHOD). [dmikurube]
- All unit tests fixed. [mikaelfrykholm]

3.0.1b2 (2012-03-01)

- For all TZID parameters in DATE-TIME properties, use timezone identifiers (e.g. Europe/Vienna) instead of timezone names (e.g. CET), as required by RFC5545. Timezone names are used together with timezone identifiers in the Timezone components. [thet]
- Timezone parsing, issues and test fixes. [mikaelfrykholm, garbas, tgecho]
- Since we use pytz for timezones, also use UTC tzinfo object from the pytz library instead of own implementation. [thet]

3.0.1b1 (2012-02-24)

- Update Release information. [thet]

3.0

- Add API for proper Timezone support. Allow creating ical DATE-TIME strings with timezone information from Python datetimes with pytz based timezone information and vice versa. [thet]
- Unify API to only use to_ical and from_ical and remove string casting as a requirement for Python 3 compatibility: New: to_ical. Old: ical, string, as_string and string casting via __str__ and str. New: from_ical. Old: from_string. [thet]

2.2 (2011-08-24)

- migration to https://github.com/collective/icalendar using svn2git preserving tags, branches and authors. [garbas]
- using tox for testing on python 2.4, 2.5, 2.6, 2.6. [garbas]
- fixed tests so they pass also under python 2.7. [garbas]
- running tests on https://jenkins.plone.org/job/icalendar (only 2.6 for now) with some other metrics (pylint, clonedigger, coverage). [garbas]
- review and merge changes from https://github.com/cozi/icalendar fork. [garbas]
- created sphinx documentation and started documenting development and goals. [garbas]
- hook out github repository to https://readthedocs.org service so sphinx documentation is generated on each commit (for master). Documentation can be visible on: https://icalendar.readthedocs.io/en/latest/ [garbas]

2.1 (2009-12-14)

- Fix deprecation warnings about `object.__init__` taking no parameters.
- Set the VALUE parameter correctly for date values.
- Long binary data would be base64 encoded with newlines, which made the iCalendar files incorrect. (This still needs testing).
- Correctly handle content lines which include newlines.

2.0.1 (2008-07-11)

- Made the tests run under Python 2.5+
- Renamed the UTC class to Utc, so it would not clash with the UTC object, since that rendered the UTC object unpicklable.

2.0 (2008-07-11)

- EXDATE and RDATE now returns a vDDDLists object, which contains a list of vDDDTypes objects. This is do that EXDATE and RDATE can contain lists of dates, as per RFC.

  Note!: This change is incompatible with earlier behavior, so if you handle EXDATE and RDATE you will need to update your code.

- When createing a vDuration of -5 hours (which in itself is nonsensical), the ical output of that was -P1DT19H, which is correct, but ugly. Now it's '-PT5H', which is prettier.

1.2 (2006-11-25)

- Fixed a string index out of range error in the new folding code.

1.1 (2006-11-23)

- Fixed a bug in caselessdicts popitem. (thanks to Michael Smith <msmith@fluendo.com>)
- The RFC 2445 was a bit unclear on how to handle line folding when it happened to be in the middle of a UTF-8 character. This has been clarified in the following discussion: http://lists.osafoundation.org/pipermail/ietf-calsify/2006-August/001126.html And this is now implemented in iCalendar. It will not fold in the middle of a UTF-8 character, but may fold in the middle of a UTF-8 composing character sequence.

1.0 (2006-08-03)

- make get_inline and set_inline support non ascii codes.
- Added support for creating a python egg distribution.

0.11 (2005-11-08)

- Changed component .from_string to use types_factory instead of hardcoding entries to 'inline'
- Changed UTC tzinfo to a singleton so the same one is used everywhere
- Made the parser more strict by using regular expressions for key name, param name and quoted/unquoted safe char as per the RFC
- Added some tests from the schooltool icalendar parser for better coverage
- Be more forgiving on the regex for folding lines
- Allow for multiple top-level components on .from_string
- Fix vWeekdays, wasn't accepting relative param (eg: -3SA vs -SA)
- vDDDTypes didn't accept negative period (eg: -P30M)
- 'N' is also acceptable as newline on content lines, per RFC

0.10 (2005-04-28)

- moved code to codespeak.net subversion.
- reorganized package structure so that source code is under 'src' directory. Non-package files remain in distribution root.
- redid doc/.py files as doc/.txt, using more modern doctest. Before they were .py files with big docstrings.
- added test.py testrunner, and tests/test_icalendar.py that picks up all doctests in source code and doc directory, and runs them, when typing:

```
python2.3 test.py
```

- renamed iCalendar to lower case package name, lowercased, de-pluralized and shorted module names, which are mostly implementation detail.
- changed tests so they generate .ics files in a temp directory, not in the structure itself.

2.2.11 Contributing

You want to help and contribute? Perfect!

These are some contribution examples

- Reporting issues to the bugtracker.
- Submitting pull requests from a forked icalendar repo.
- Extending the documentation.
- Sponsor a Sprint (https://plone.org/events/sprints/whatis).

For pull requests, keep this in mind

- Add a test which proves your fix and make it pass.
- Describe your change in CHANGES.rst
- Add yourself to the docs/credits.rst

Development Setup

If you would like to setup icalendar to contribute changes, the Installation Section should help you further.

PYTHON MODULE INDEX

i

INDEX

W