

datemultical Package

Named dates with multi-calendar formatting in $\text{\LaTeX}$

Amer Amikhteh

Version 0.2.0
October 18, 2026

Abstract

The `datemultical` package provides a unified interface for defining, converting, and formatting named dates in $\text{\LaTeX}$ documents. Its basic backend supports the Gregorian, Solar Hijri, and Lunar Hijri calendars under all supported engines, with English names and optional Persian localization.

Under $\text{\LuaTeX}$, the Lua backend is selected by default. It supports all calendar systems provided by `lua-calendrica` (currently 54, including the three core calendars and a range of historical and astronomical calendars). Newly added calendars become available automatically, without updating `datemultical`; the basic backend can be selected with the `native` option.

1 Package Architecture and Backend Selection

`datemultical` provides two calculation backends. The basic backend is the built-in implementation and supports the three core calendars—Gregorian, Solar Hijri, and Lunar Hijri—under all supported engines. The Lua backend is an extended implementation of this core functionality: it includes the same three core calendars and additionally supports the calendar systems provided by `lua-calendrica`.

Table 1: Backend Scope and Engine Availability

Backend	Scope	Engine availability
<code>basic</code>	Three core calendars: Gregorian, Solar Hijri, and Lunar Hijri	All supported engines
<code>lua</code>	The three core calendars together with the additional calendar systems provided by <code>lua-calendrica</code>	$\text{\LuaTeX}$ only

Under $\text{\LuaTeX}$, the Lua backend is selected by default. The `native` option selects the basic backend instead. Under non-Lua engines, the basic backend is selected automatically.

The package has two options:

- **native**: selects the basic backend. It is particularly relevant under Lua $\LaTeX$, where the Lua backend would otherwise be selected by default.
- **persian**: enables Persian-localization features. This option is independent of backend selection.

Thus, **native** is an option name, whereas **basic** and **lua** are backend names. The Lua backend contains the core functionality of the basic backend but is available only under Lua $\LaTeX$.

Table 2: Core Calendar Identifiers and Lua Mappings

Identifier (<i>c</i>)	Calendar System	Backend	Lua Mapping
<code>gr</code>	Gregorian	<code>basic</code>	<code>gregorian</code>
<code>sh</code>	Solar Hijri	<code>basic</code>	<code>persian</code>
<code>lh</code>	Lunar Hijri	<code>basic</code>	<code>islamic-tbla</code>

Beyond the three core identifiers, the Lua backend routes conversions through the `lua-calendrica` modules, enabling access to numerous historical and astronomical calendar systems under Lua $\LaTeX$.

2 Core Public API

2.1 Argument Specifications

Table 3: Standard Argument Specifications for Package Commands

Arg	Type	Name	Description / Allowed Values
<code>c</code>	string	Calendar	Calendar key (<code>gr</code> , <code>sh</code> , <code>lh</code> , ...); optional, default: <code>gr</code> .
<code>n</code>	string	Name	Unique identifier for the date instance (e.g., <code>birth</code>).
<code>y</code>	integer	Year	Astronomical/civil year (e.g., <code>1365</code>).
<code>m</code>	integer	Month	Month number (1–12).
<code>d</code>	integer	Day	Day of the month (1–31).
<code>f</code>	string	Field	Target property: <code>year</code> , <code>month</code> , <code>day</code> , <code>jdn</code> , <code>weekday</code> .
<code>p</code>	string	Pattern	Format pattern string using defined tokens (e.g., <code>yyyy-MM-dd</code>).
<code>s</code>	string	Style	Registered style preset identifier (e.g., <code>full</code> , <code>short</code> , <code>ISOext</code>).

2.2 Command Reference

Table 4: Core Public Command Reference

Syntax	Function	Example	Output
<code>\dmcset[c]{n}{y}{m}{d}</code>	Define date	<code>\dmcset[sh]{birth}{1365}{3}{11}</code>	
<code>\dmcget[c]{n}{f}</code>	Get field	<code>\dmcget[gr]{birth}{year}</code>	1986
<code>\dmcformat[c]{n}{p}</code>	Custom format	<code>\dmcformat[gr]{birth}{d~MMMM~y}</code>	1 June 1986
<code>\dmcstyle{s}{p}</code>	Register style	<code>\dmcstyle{iso}{yyyy-MM-dd}</code>	
<code>\dmc[c]{n}{s}</code>	Predefined style	<code>\dmc[gr]{birth}{short}</code>	1986/06/01

2.3 dmcnew command and keys

The command `\dmcnew` is available only with Lua¹TeX. It creates and stores a named date using calendar-specific key–value pairs.

```
\dmcnew[calendar]{name}{key=value, ...}
```

Table 5: Keys associated with `\dmcnew`

Key	Value	Definition	Backend
<i>Basic date components</i>			
year	integer	Year within the calendar.	basic
month	integer	Month within the year or calendar cycle.	basic
day	integer	Day within the month or calendar cycle.	basic
<i>Derived date properties</i>			
weekday	integer	Number of the weekday of the date.	basic
jdn	integer	Absolute day number in the Julian Day system.	basic
rd	integer	Absolute day number in the Rata Die system.	basic
<i>Calendar cycle components</i>			
cycle	integer	Number of the calendar’s repeating cycle.	lua
week	integer	Week number within the calendar year or season.	lua
major	integer	Number of the larger cycle containing the calendar year.	lua
<i>Intercalary date components</i>			
leap	boolean	Whether the date belongs to a leap year.	lua
leap_month	boolean	Whether the date belongs to an intercalary month.	lua
leap_day	boolean	Whether the date is an intercalary day.	lua
<i>Calendar-specific components</i>			
era	string	Era in which the date occurs.	lua
season	string	Season containing the date.	lua

Continued on the next page

Key	Value	Definition	Backend
event	string	Named calendrical event associated with the date.	lua
count	integer	Day count relative to the named calendrical event.	lua
prefix	string	Prefix of an Akan day name.	lua
stem	string	Stem of an Akan day name.	lua
number	integer	Number in a named sacred day cycle.	lua
name	string	Name in a named sacred day cycle.	lua
<i>Maya Long Count components</i>			
baktun	integer	Baktun position in the Long Count.	lua
katun	integer	Katun position in the Long Count.	lua
tun	integer	Tun position in the Long Count.	lua
uinal	integer	Uinal position in the Long Count.	lua
kin	integer	Kin position in the Long Count.	lua
<i>Balinese Pawukon components</i>			
luang	boolean	Whether the date has the luang designation.	lua
dwiwara	integer	Position in the two-day Pawukon cycle.	lua
triwara	integer	Position in the three-day Pawukon cycle.	lua
caturwara	integer	Position in the four-day Pawukon cycle.	lua
pancawara	integer	Position in the five-day Pawukon cycle.	lua
sadwara	integer	Position in the six-day Pawukon cycle.	lua
saptawara	integer	Position in the seven-day Pawukon cycle.	lua
asatawara	integer	Position in the eight-day Pawukon cycle.	lua
sangawara	integer	Position in the nine-day Pawukon cycle.	lua
dasawara	integer	Position in the ten-day Pawukon cycle.	lua

Example 1: Chinese calendar

```
\dmcnew[chinese]{birth}{
  cycle=78,year=3,month=4,leap=false,day=24}
```

This example defines a date in the traditional Chinese lunisolar calendar. The year is specified as year 3 of cycle 78, and the date falls on day 24 of month 4. The Boolean key `leap` indicates whether the month is intercalary.

Example 2: Hindu lunar calendar

```
\dmcnew[hindu-lunar]{birth}{
  year=2043, month=2, leap_month=false, day=24,
  ↪ leap_day=false}
```

This example defines a date in the modern Hindu lunisolar calendar. In addition to the year, month, and day, it uses the `leap_month` and `leap_day` keys. These Boolean keys indicate whether the month or the day is intercalary, respectively.

3 Date Formatting

3.1 Format Pattern Tokens

The following format tokens are supported by `\dmcformat` and `\dmcstyle`:

- **Year:** `yyyy` (four-digit year), `yy` (two-digit year), `y` (numeric year).
- **Month:** `MMMM` (full name), `MMM` (abbreviated name), `MMMMP` (Persian name), `MM` (two-digit number), `M` (numeric month).
- **Day:** `dd` (two-digit number), `d` (numeric day).
- **Weekday:** `EEEE` (full name), `EEE` (abbreviated name), `EEEEP` (Persian name), `U` (numeric weekday).
- **Week:** `ww` (two-digit week), `w` (numeric week).
- **Calendar cycles:** `U` (cycle number), `R` (major cycle number).
- **Leap values:** `l` (leap year), `li` (leap-year flag), `lm` (leap month), `lmi` (leap-month flag), `ld` (leap day), `ldi` (leap-day flag), `lu` (luang), `lui` (luang flag).
The tokens `li`, `lmi`, `ldi`, and `lui` convert the corresponding Boolean value to `1` or `0`.
- **Calendar properties:** `S` (season), `G` (era), `Gi` (numeric era identifier), `E` (event), `C` (event count).
- **Named calendar values:** `N` (name), `n` (number), `p` (prefix), `s` (stem).
- **Maya Long Count:** `bt` (baktun), `ka` (katun), `tn` (tun), `ui` (uinal), `ki` (kin).
- **Balinese Pawukon cycles:** `di` (dwiwara), `tr` (triwara), `cw` (caturwara), `pw` (pancawara), `sw` (sadwara), `sp` (saptawara), `aw` (asatawara), `sg` (sangawara), `da` (dasawara).

Tokens are replaced only when the corresponding value is defined for the target calendar. Unrecognized tokens remain unchanged.

3.2 Predefined Style Presets

The package provides standard presets out of the box (evaluated below for `birth` in `gr`):

Table 6: Standard Style Presets and Evaluated Outputs

Style	Pattern	Invocation Example	Evaluated Output
full	EEEE,~MMMM~d,~y	<code>\dmc{birth}{full}</code>	Sunday, June 1, 1986
long	MMMM~d,~y	<code>\dmc{birth}{long}</code>	June 1, 1986
medium	MMM~d,~y	<code>\dmc{birth}{medium}</code>	Jun 1, 1986
short	yy/MM/dd	<code>\dmc{birth}{short}</code>	1986/06/01
short-use	M/d/yy	<code>\dmc{birth}{short-use}</code>	6/1/1986
ISO	yyMMdd	<code>\dmc{birth}{ISO}</code>	19860601
ISOext	yy-MM-dd	<code>\dmc{birth}{ISOext}</code>	1986-06-01
american	MMMM~dd,~yy	<code>\dmc{birth}{american}</code>	June 01, 1986
shamerican	MM/dd/yy	<code>\dmc{birth}{shamerican}</code>	06/01/1986
british	dd~MMMM~yy	<code>\dmc{birth}{british}</code>	01 June 1986
shbritish	dd/MM/yy	<code>\dmc{birth}{shbritish}</code>	01/06/1986
shbritishdots	dd.MM.yy	<code>\dmc{birth}{shbritishdots}</code>	01.06.1986

Existing styles can be overridden at any point by simply redefining them with the same command and new pattern. The package uses a direct property-list replacement mechanism, so no error occurs and the change applies globally. Example:

```
\dmcstyle{full}{EEE,~MMMM~d,~y}
```

3.3 Persian Date Wrappers

Pre-configured high-level macros provide convenient date rendering with appropriate text directionality:

Table 7: High-Level Directional Formatting Wrappers

Macro Command	c	Effective Pattern	Language	Output
<code>\Miladi{n}</code>	gr	MMMM~d,~y	English	June 1, 1986
<code>\Shamsi{n}</code>	sh	d~MMMMP~y	Persian	۱۳۶۵ خرداد ۱۱
<code>\Ghamari{n}</code>	lh	d~MMMMP~y	Persian	۱۴۰۶ رمضان ۲۴
<code>\ShortMiladi{n}</code>	gr	yy/MM/dd	English	1986/06/01
<code>\ShortShamsi{n}</code>	sh	yy/MM/dd	Persian	۱۳۶۵/۰۳/۱۱
<code>\ShortGhamari{n}</code>	lh	yy/MM/dd	Persian	۱۴۰۶/۰۹/۲۴
<code>\FaMiladi{n}</code>	gr	d~MMMMP~y	Persian	۱۹۸۶ ژوئن ۱
<code>\Persianweekday{n}</code>	sh	d~EEEEP~y	Persian	یکشنبه

When using `xepersian` or `unipersian`, directional switching (`\rl` / `\lr`) is handled automatically by the respective package. `datemultical` only provides high-level formatting wrappers that are fully compatible with these packages.

4 Lua Calendar Reference

Under LuaLaTeX, `datemultical` supports all 54 calendar modules provided by version 0.2 of `lua-calendrica`. The following table lists the calendar identifiers, engine keys (*c*), and sample outputs.

Table 8: Lua Calendar Modules, Identifiers, and Sample Outputs

No.	Identifier (<i>c</i>)	Output
1	akan	3/5
2	amazigh	2936/05/19
3	armenian	1435/11/06
4	auc	2739/05/19
5	aztec-tonalpohualli	2/20
6	aztec-xihuitl	12/12
7	babylonian	2297/02/0/22
8	bahai	1/8/10/04/16
9	bahai-astronomical	1/8/10/04/16
10	balinese	1/2/1/2/4/4/1/2/1/0
11	bengali-revised	1393/02/18
12	chinese	78/03/04/0/24
13	coptic	1702/09/24
14	egyptian	2735/02/06
15	ethiopic	1978/09/24
16	french	194/09/12
17	french-arithmetic	194/09/12
18	gregorian	1986/06/01
19	hebrew	5746/02/23
20	hebrew-civil	5746/09/23
21	hebrew-observational	5746/03/22
22	hindu-lunar	2043/02/0/24/0
23	hindu-lunar-astronomical	2043/02/0/24/0
24	hindu-solar	1908/02/19
25	hindu-solar-astronomical	1908/02/18
26	icelandic	1986/90/06/0
27	indian-national	1908/03/11
28	islamic	1406/09/23
29	islamic-diyanet	1406/09/23
30	islamic-observational	1406/09/22
31	islamic-saudi	1406/09/23
32	islamic-tbla	1406/09/24

Continued on the next page

No.	Identifier (c)	Output
33	islamic-turkish	1406/09/23
34	islamic-umalqura	1406/09/23
35	iso	1986/22/07
36	japanese-era	3/61/06/01
37	japanese	78/03/04/0/24
38	julian	1986/05/19
39	korean	4319/04/0/24
40	mayan-long-count	12/18/13/1/0
41	mayan-haab	03/13
42	mayan-tzolkin	2/20
43	old-hindu-lunar	5087/02/0/24
44	old-hindu-solar	5087/02/16
45	olympiad	691/02/05/19
46	persian-arithmetic	1365/03/11
47	persian	1365/03/11
48	roc	75/06/01
49	roman	1986/06/1/14/0
50	samaritan	3624/02/24
51	thai-solar	2529/06/01
52	tibetan	2113/04/0/24/0
53	nanakshahi	518/03/18
54	vietnamese	78/03/04/0/24

Auxiliary Modules: In addition to the conversion modules above, `lua-calendrica` includes core utilities for astronomical computations, general calendrical arithmetic, and ecclesiastical feast calculations.

For a complete description of the calendar modules provided by `lua-calendrica`, please consult the following documentation:

<https://polyglossia.codeberg.page/lua-calendrica/topics/calendars.md.html>

I am grateful to Udi Fogiel for developing and maintaining `lua-calendrica`, and for suggesting automatic calendar discovery.

5 Current Limitations and Possible Directions

The following items are possible directions rather than committed development plans.

- Localization is currently available only in Persian. Support for additional languages could be considered if future work on the package is undertaken.
- Improved interoperability with other date-related $\text{\TeX}$ packages, such as `datetime2`, could be another useful direction.