

datemultical Package

Named dates with multi-calendar formatting in $\LaTeX$

Amer Amikhteh

Version 0.1.0
September 17, 2026

Abstract

The `datemultical` package provides a unified interface for defining, converting, and formatting dates in $\LaTeX$ documents. Its basic backend implements three core calendars—Gregorian, Solar Hijri, and Lunar Hijri—under all supported engines. This includes date conversion and formatting with English month names and weekday names, while Persian names are available when Persian localization is enabled.

Under $\text{Lua}\LaTeX$, the Lua backend is selected by default. The Lua backend is an extended implementation of the same core functionality: it includes the three core calendars implemented by the basic backend and, through `lua-calendrica`, supports 39 calendar systems in total, including historical and astronomical calendars. The basic backend can be selected under $\text{Lua}\LaTeX$ with the `native` option. This guide describes the package architecture, public API, calendar identifiers, formatting patterns and styles, Persian date and weekday wrappers, and Lua calendar support.

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>	$\text{Lua}\LaTeX$ only

Under $\text{Lua}\LaTeX$, 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 $\LaTeX$.

2 Core Calendar Identifiers

Table 2: Core Calendar Identifiers and Lua Mappings

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

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 $\LaTeX$.

3 Core Public API

3.1 Argument Specifications

Table 3: Standard Argument Specifications for Package Commands

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

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

3.2 Command Reference

3.3 dmcnew command and keys

The command `\dmcnew` is available only with Lua^ATeX. 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	Use	Backend
<code>year</code>	integer	Calendar year	<code>basic</code>
<code>month</code>	integer	Calendar month	<code>basic</code>
<code>day</code>	integer	Day of the month	<code>basic</code>
<code>weekday</code>	integer	Day of the week	<code>basic</code>
<code>jdn</code>	integer	Julian Day Number	<code>basic</code>
<code>rd</code>	integer	Rata Die	<code>basic</code>
<code>week</code>	integer	Week number	<code>lua</code>
<code>cycle</code>	integer	Calendar cycle	<code>lua</code>
<code>major</code>	integer	Calendar-specific major cycle	<code>lua</code>
<code>leap</code>	boolean	Leap flag	<code>lua</code>
<code>leap_month</code>	boolean	Leap-month flag	<code>lua</code>
<code>leap_day</code>	boolean	Leap-day flag	<code>lua</code>

Example:

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

4 Date Formatting

4.1 Format Pattern Tokens

Standard formatting tokens (following Unicode Technical Standard #35 conventions) supported by `\dmcformat` and `\dmcstyle`:

- **Year:** yyyy (full 4-digit year), yy (2-digit year), y (numeric year).
- **Month:** MM (2-digit zero-padded), M (numeric), MMM (abbreviated), MMMM (full), MMMMP (localized Persian month name).
- **Day:** dd (2-digit zero-padded), d (numeric).
- **Weekday:** EEEE (full weekday name), EEE (abbreviated), EEEEEP (localized Persian weekday).
- **Week:** ww (2-digit zero-padded week), w (numeric week).
- **Continuum Cycle:** U (cycle within major era).
- **Major Cycle:** R (major cycle/era identifier).
- **Leap information:** l/Lm/Ld refer to leap/leap_month/leap_day; the suffix i returns the Boolean value as 1 or 0.

All tokens are replaced intelligently and only if they are defined in the target calendar. Unrecognized tokens are left unchanged.

4.2 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}
```

4.3 Persian Date Wrappers

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

When using `xepersian` or `unipersian`, directional switching (`\rl` / `\lr`) is han-

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

dled automatically by the respective package. `datemultical` only provides high-level formatting wrappers that are fully compatible with these packages.

5 Lua Calendar Reference

Under Lua^LTEX, `datemultical` interfaces with `lua-calendrica` (based on Reingold & Dershowitz, *Calendrical Calculations*). Below is the mapping of supported calendar modules, engine keys (*c*), and aliases:

Identifier (<i>c</i>)	Output
armenian	1435/11/06
auc	2739/05/19
aztec_xihuitl	12/12
babylonian	2297/02/0/22
bahai	1/8/10/04/16
bahai_astronomical	1/8/10/04/16
chinese	78/03/04/0/24
coptic	1702/09/24
egyptian	2735/02/06
ethiopic	1978/09/24
french	194/09/12
french_arithmetic	194/09/12
gregorian	1986/06/01
hebrew	5746/02/23
hebrew_observational	5746/03/22
hindu_lunar	2043/02/0/24/0
hindu_lunar_astronomical	2043/02/0/25/0
hindu_solar	1908/02/19
hindu_solar_astronomical	1908/02/18

Continued on the next page

Identifier (<i>c</i>)	Output
islamic	1406/09/23
islamic_observational	1406/09/22
islamic_saudi	1406/09/23
islamic_tbla	1406/09/24
islamic_turkish	1406/09/22
islamic_umatqura	1406/09/23
islamic_umatqura_astronomical	1406/09/23
iso	1986/22/07
japanese	78/03/04/0/24
julian	1986/05/19
korean	4319/04/0/24
mayan_haab	03/13
old_hindu_lunar	5087/02/0/24
old_hindu_solar	5087/02/16
olympiad	691/02/05/19
persian	1365/03/11
persian_arithmetic	1365/03/11
samaritan	3624/02/24
tibetan	2113/04/0/24/0
vietnamese	03/04/24

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

6 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.
- Closer coordination with `lua-calendrica` could provide access to additional calendar systems and make better use of the capabilities offered by that package.
- Improved interoperability with other date-related $\text{\TeX}$ packages, such as `datetime2`, could be another useful direction.