INTERNATIONAL STANDARD

ISO 8601

The international standard for representing dates and times in an unambiguous, machine-readable format.

Whether you're building APIs, storing timestamps in databases, or exchanging data between systems, ISO 8601 ensures everyone interprets dates and times the same way—regardless of locale, language, or timezone.

UNAMBIGUOUS
No confusion between MM/DD and DD/MM formats

SORTABLE
Alphabetical sort equals chronological sort

UNIVERSAL
Works across all languages and regions

PRECISE
Includes timezone information when needed

CURRENT EDITION
ISO 8601-1:2019
Part 1: Basic rules

EXTENSION
ISO 8601-2:2019
Part 2: Extensions

FIRST PUBLISHED
1988
ISO 8601:1988

▸ CURRENT UTC TIME IN ISO 8601 FORMAT

YEAR	MONTH	DAY	HOUR	MIN	SEC
2026	10	29	08	40	03

2026-10-29T08:40:03Z_

CALENDAR DATE	TIME (UTC)	WEEK DATE	ORDINAL DATE
2026-10-29	08:40:03Z	2026-W44-4	2026-302

INTERACTIVE

FORMAT BUILDER

Configure your ISO 8601 format and see it applied to the current time

DATE FORMAT

Calendar (YYYY-MM-DD) | Ordinal (YYYY-DDD) | Week (YYYY-Www-D)

TIME PRECISION

Date only | Hours | Minutes | Seconds | Milliseconds

TIMEZONE

UTC (Z) | Local offset | No timezone

SEPARATOR STYLE

Extended (with -/:) | Basic (compact)

LIVE OUTPUT

2026-10-29T08:40:03Z

COPY FORMAT TO CLIPBOARD

FORMAT PATTERN

YYYY-MM-DDThh:mm:ssZ

FORMAT SUMMARY

Complete overview of ISO 8601 date and time representations

ISO 8601 defines two format styles: **basic** (compact, no separators) and **extended** (human-readable, with separators). Both are valid and can represent the same information. The extended format is more common in APIs and user interfaces, while the basic format saves space in storage and transmission.

CALENDAR DATE	
BASIC YYYYMMDD **20261029**	EXTENDED YYYY-MM-DD **2026-10-29**

ORDINAL DATE	
BASIC YYYYDDD **2026302**	EXTENDED YYYY-DDD **2026-302**

WEEK DATE	
BASIC YYYYWwwD **2026W444**	EXTENDED YYYY-Www-D **2026-W44-4**

LOCAL TIME	
BASIC hhmmss **084003**	EXTENDED hh:mm:ss **08:40:03**

UTC TIME	
BASIC hhmmssZ **084003Z**	EXTENDED hh:mm:ssZ **08:40:03Z**

TIME WITH OFFSET	
BASIC hhmmss±hhmm **084003+0100**	EXTENDED hh:mm:ss±hh:mm **08:40:03+01:00**

DATE AND TIME	
BASIC YYYYMMDDThhmmssZ **20261029T084003Z**	EXTENDED YYYY-MM-DDThh:mm:ssZ **2026-10-29T08:40:03Z**

DURATION	
BASIC PnYnMnDTnHnMnS **P3Y6M4DT12H30M5S**	EXTENDED PnYnMnDTnHnMnS **P3Y6M4DT12H30M5S**

TIME INTERVAL	
BASIC start/end **20260101/20261231**	EXTENDED start/end **2026-01-01/2026-12-31**

▸ WHEN TO USE BASIC FORMAT

Use basic format when storage space or bandwidth matters, such as in filenames, log files, or compact data serialization. Fewer characters, same information.

▸ WHEN TO USE EXTENDED FORMAT

Use extended format for APIs, databases, and anywhere humans need to read the data. The separators make it much easier to parse visually.

SECTION 02

CALENDAR DATE

Representation of dates using the Gregorian calendar

The calendar date format is the most commonly used date representation. It follows a **year-month-day** order, which is unambiguous regardless of locale. Unlike regional formats (12/02/2024 could mean December 2nd or February 12th), ISO 8601 dates have only one possible interpretation.

COMPLETE REPRESENTATION

FORMAT

YYYY-MM-DD

YYYY	Four-digit year (0000-9999)
MM	Two-digit month (01-12)
DD	Two-digit day of month (01-31)

Tip: Always use leading zeros. Write "03" not "3" for March, and "09" not "9" for the 9th day.

LIVE EXAMPLE

October 29, 2026	**2026-10-29**
January 1, 2000	**2000-01-01**
July 20, 1969	**1969-07-20**
February 29, 2024	**2024-02-29**

TRUNCATED REPRESENTATIONS

When full precision isn't needed, you can omit smaller units:

Year and month only	2026-10
Year only	2026
Century only	20

WHY THIS ORDER?

ISO 8601 uses **big-endian** order (largest unit first), similar to how we write numbers. Just as 1234 means one thousand, not four thousand, 2024-12-02 unambiguously means year 2024, month 12, day 02. This order also enables simple alphabetical sorting to produce chronological order.

SECTION 03

TIME OF DAY

24-hour clock representations for time values

ISO 8601 uses the **24-hour clock** (also called military time), eliminating the ambiguity of AM/PM. Midnight is 00:00:00, noon is 12:00:00, and 11 PM is 23:00:00. Times are written from largest unit (hours) to smallest (seconds or fractions).

LOCAL TIME

FORMAT

hh:mm:ss

hh	Two-digit hour (00-24)
mm	Two-digit minute (00-59)
ss	Two-digit second (00-60*)

* Value 60 is permitted only for positive leap second representation

DECIMAL FRACTIONS

For sub-second precision, add a decimal fraction to the smallest unit. ISO 8601 prefers comma as the decimal separator, but period is also accepted and more common in programming.

With fractional seconds	**08:40:03,9**
Fractional minutes	**08:40,5**
Fractional hours	**08,5**

MIDNIGHT

00:00:00 — Beginning of a calendar day

24:00:00 — End of a calendar day (same instant as next day's 00:00:00)

REDUCED PRECISION

Omit seconds or minutes when not needed:

Hours and minutes	08:40
Hours only	08

BASIC FORMAT

Compact form without colons:

Complete	084003
Reduced	0840

DATE-TIME

Combine date and time with T separator:

2026-10-29T08:40:03

INTERACTIVE

TIME BUILDER

Configure your time format options and see the live result

PRECISION

Hours | Minutes | Seconds | Milliseconds

TIMEZONE

Local (none) | UTC (Z) | Offset (±hh:mm)

DECIMAL SEPARATOR

Period (.) | Comma (,)

FORMAT STYLE

Extended (hh:mm:ss) | Basic (hhmmss)

LIVE OUTPUT

08:40:03Z

COPY FORMAT TO CLIPBOARD

FORMAT PATTERN

hh:mm:ssZ

The Z suffix (from "Zulu" in NATO phonetic alphabet) explicitly indicates UTC.

SECTION 04

TIME ZONES & UTC

Coordinated Universal Time and timezone offset designators

A time without timezone information is ambiguous—"3 PM" means different moments depending on where you are. ISO 8601 solves this by appending a **timezone designator** that specifies the offset from UTC. For timestamps that need to be unambiguous globally (like API responses or log entries), always include the timezone.

UTC DESIGNATOR

Z

Zulu Time
Zero UTC offset

The suffix **Z** (from "Zulu" in the NATO phonetic alphabet) indicates UTC. It's equivalent to +00:00 but more concise. Use Z for UTC timestamps in APIs and databases.

OFFSET FORMAT

FORMAT

±hh:mm

± Plus (+) for east of UTC, minus (-) for west

hh Hours offset (00-14)

mm Minutes offset (00-59)*

* In practice, only 00, 30, and 45 are used by current time zones

LIVE OFFSET EXAMPLES

The same instant in time, represented with different timezone offsets:

ZONE	OFFSET	08:40:03 UTC =
UTC	Z	**08:40:03Z**
UTC+0	+00:00	**08:40:03+00:00**

EST (UTC-5)	-05:00	**03:40:03-05:00**
IST (UTC+5:30)	+05:30	**14:10:03+05:30**
JST (UTC+9)	+09:00	**17:40:03+09:00**
NZST (UTC+12)	+12:00	**20:40:03+12:00**

▸ OFFSET VS TIMEZONE NAME

ISO 8601 uses numeric offsets, not timezone names like "EST" or "PST". This is intentional—offsets are unambiguous, while names can be ambiguous (CST could be Central Standard Time or China Standard Time) and change with daylight saving.

▸ BEST PRACTICE

Store and transmit timestamps in UTC (with Z suffix), then convert to local time for display. This avoids daylight saving complications and makes comparison straightforward.

SECTION 05

WEEK DATE

Representation using ISO week-numbering year and week number

Week dates are useful when you care more about "which week" than "which month"—common in business planning, project scheduling, and payroll systems. The ISO week system ensures every week has exactly 7 days and belongs to exactly one year, avoiding the split weeks that occur at month boundaries.

WEEK DATE FORMAT

FORMAT

YYYY-Www-D

YYYY	ISO week-numbering year (may differ from calendar year)
W	Week designator (literal letter W)
ww	Week number (01-53)
D	Day of week (1=Monday through 7=Sunday)

ISO WEEK RULES

The ISO week system has specific rules that differ from some regional conventions:

- **Weeks start on Monday** (not Sunday)
- Week 01 contains the **first Thursday** of the year
- Equivalently, Week 01 contains **January 4th**
- Years have either **52 or 53 weeks**

LIVE EXAMPLE

2026-10-29 (Thu)	**2026-W44-4**
2026-01-01 (Wed)	**2026-W01-3**
2020-12-31 (Thu)	**2020-W53-4**

> **YEAR BOUNDARY GOTCHA**
>
> The ISO week-numbering year may differ from the calendar year near December/January. For example, December 30, 2024 (Monday) is actually 2025-W01-1 because that week contains the first Thursday of 2025. Conversely, January 1, 2021 was 2020-W53-5. Always use the ISO week year, not the calendar year, when working with week dates.

SECTION 06

ORDINAL DATE

Representation using year and day-of-year number

Ordinal dates express a date as the day number within a year (1-365 or 1-366 for leap years). This format is compact and makes it easy to calculate the number of days between dates within the same year. It's widely used in astronomy, aviation, and military contexts where day-counting simplifies operations.

ORDINAL DATE FORMAT

LIVE EXAMPLE

2026-302

Today is day 302 of 2026

YYYY	Four-digit year
DDD	Three-digit day of year (001-366)

COMMON USES

- **Astronomy:** Julian dates and star catalogs
- **Aviation:** Flight planning and maintenance logs
- **Manufacturing:** Batch and lot numbering
- **Military:** Operations planning (often called "Julian dates")

CONVERSION TABLE (2026)

Converting between calendar and ordinal dates for the current year:

DATE	ORDINAL	DAY #
Jan 1	2026-001	1
Feb 1	2026-032	32
Jul 4	2026-185	185
Today	**2026-302**	**302**
Dec 31	2026-365	365

2026 is not a leap year (365 days)

▸ NOT THE SAME AS JULIAN DATE

Although often called "Julian dates" colloquially, ISO 8601 ordinal dates are different from astronomical Julian dates. The astronomical Julian date is a continuous count of days since January 1, 4713 BC, used primarily in astronomy. ISO 8601 ordinal dates restart at 001 each year.

SECTION 07

DURATION

Representation of spans of time

Durations represent a **length of time** rather than a specific point in time. They're useful for expressing things like "3 hours", "2 weeks", or "1 year, 6 months, and 15 days". All durations begin with the letter **P** (for "period"), and time components are separated from date components by **T**.

DURATION FORMAT

FORMAT

PnYnMnDTnHnMnS

P	Period designator (required, starts every duration)
nY	Number of years
nM	Number of months (before T)
nD	Number of days
T	Time designator (required before hours/minutes/seconds)
nH	Number of hours
nM	Number of minutes (after T)
nS	Number of seconds (can include decimals)

EXAMPLES

Only include the units you need. Omit any with zero value:

3 years	**P3Y**
6 months, 15 days	**P6M15D**
4 hours, 30 minutes	**PT4H30M**
1 year, 2 months, 3 days, 4 hours	**P1Y2M3DT4H**
90 seconds	**PT90S**

0.5 seconds	**PT0.5S**

WEEK DURATION

Weeks can only be used alone—you cannot mix weeks with other units:

2 weeks	**P2W**
Invalid:	~~P1M2W~~

▸ **COMMON CONFUSION**

Note that M means different things before and after the T: P2M is 2 months, but PT2M is 2 minutes. The T separator is what distinguishes them. Similarly, P1D is 1 day, while PT24H is 24 hours (which may not equal 1 day during daylight saving transitions).

INTERACTIVE

DURATION BUILDER

Set the values for each unit to build your ISO 8601 duration string

Date/Time Units | Weeks Only

DATE COMPONENTS

0 years | 0 months | 0 days

TIME COMPONENTS

0 hours | 0 minutes | 0 seconds

Reset All

LIVE OUTPUT

0 days

COPY FORMAT TO CLIPBOARD

ISO 8601 DURATION

P0D

SECTION 08

TIME INTERVALS

Representation of time periods between two points

Intervals represent a span of time—useful for expressing date ranges, booking periods, or subscription terms. ISO 8601 supports four ways to express an interval: start/end, start/duration, duration/end, or duration alone. All use a forward slash (/) as the separator.

START AND END

When you know both the start and end dates:

FORMAT
<start>/<end>

2026-01-01/2026-12-31

Full year 2026

START AND DURATION

When you know when it starts and how long it lasts:

FORMAT
<start>/<duration>

2026-10-29/P1M

One month starting today

DURATION AND END

When you know when it ends and how long it lasts:

FORMAT
<duration>/<end>

P6M/2026-12-31

6 months ending Dec 31, 2026

DURATION ONLY

When context provides the start or end point:

FORMAT
<duration>

P1Y2M

1 year and 2 months (context-dependent)

RECURRING INTERVALS

For events that repeat, prefix the interval with R followed by an optional count. If no count is given, the recurrence is unbounded (repeats forever).

FORMAT

Rn/interval

Components

R	Recurring designator (required)
n	Repeat count (optional—omit for infinite)

Examples

R5/2026-10-29/P1M
5 monthly occurrences starting today

R/PT1H
Repeat every hour, indefinitely

R12/2026-01-01/P1M
Monthly for a full year

▸ INCLUSIVE VS EXCLUSIVE

ISO 8601 doesn't specify whether interval endpoints are inclusive or exclusive. Convention varies by application—document your choice clearly when it matters.

▸ COMBINING WITH TIME

Intervals can use full datetime values, not just dates. For example: 2026-10-29T09:00:00Z/2026-10-29T17:00:00Z represents working hours.

SECTION 09

HISTORY & ADOPTION

Development timeline and global implementation

EDITION TIMELINE

1988	First edition published as ISO 8601:1988
1991	ISO 8601:1988/Cor 1:1991 — Technical corrigendum
2000	Second edition: ISO 8601:2000 — Revised and expanded
2004	Third edition: ISO 8601:2004 — Further revisions

2019	Fourth edition split into two parts: ISO 8601-1:2019 (Basic rules) and ISO 8601-2:2019 (Extensions)

RELATED STANDARDS

RFC 3339
Internet date/time format (profile of ISO 8601)

ISO 8601-2:2019
Extensions (uncertain dates, seasons, etc.)

W3C NOTE-datetime
W3C profile for web applications

EDTF
Extended Date/Time Format (Library of Congress)

ADOPTION

ISO 8601 has been adopted as a national standard in numerous countries and is mandated or recommended by:

European Union	United Nations
NATO	IETF
W3C	Most APIs

▸ **XKCD ENDORSEMENT**

The ISO 8601 format has achieved notable cultural recognition, including endorsement by the webcomic XKCD in comic #1179, which humorously advocates for the standard as the only sensible date format for avoiding ambiguity.

SECTION 10

PROGRAMMING REFERENCE

How to format and parse ISO 8601 dates in popular languages

Examples show common ISO 8601 formatting operations including full datetime, date only, time only, week dates, durations, and parsing. Output examples show typical values for December 2, 2024 at 14:30:00 UTC.

JAVASCRIPT

FULL ISO DATETIME (UTC)
```
new Date().toISOString()
```
→ 2024-12-02T14:30:00.000Z

DATE ONLY
```
new Date().toISOString().split('T')[0]
```
→ 2024-12-02

LOCAL DATETIME
```
new Date().toLocaleString('sv-SE')
```
→ 2024-12-02 14:30:00

CUSTOM FORMAT
```
date.toISOString().replace('T', ' ').slice(0, 19)
```
→ 2024-12-02 14:30:00

PARSE ISO STRING
```
new Date('2024-12-02T14:30:00Z')
```

UNIX TIMESTAMP TO ISO

PYTHON

FULL ISO DATETIME (UTC)
```
datetime.now(UTC).isoformat()
```
→ 2024-12-02T14:30:00+00:00

DATE ONLY
```
date.today().isoformat()
```
→ 2024-12-02

TIME ONLY
```
datetime.now().time().isoformat()
```
→ 14:30:00.000000

WITHOUT MICROSECONDS
```
datetime.now().isoformat(timespec='seconds')
```
→ 2024-12-02T14:30:00

WEEK NUMBER
```
date.today().isocalendar()
```
→ (2024, 49, 1)

PARSE ISO STRING

```
new Date(1701528600000).toISOString()
```

```
datetime.fromisoformat('2024-12-02T14:30:00')
```

DURATION (TIMEDELTA)

```
isodate.duration_isoformat(timedelta(hours=2))
→ PT2H
```

JAVA

FULL ISO DATETIME (UTC)

```
Instant.now().toString()
→ 2024-12-02T14:30:00.000Z
```

DATE ONLY

```
LocalDate.now().toString()
→ 2024-12-02
```

TIME ONLY

```
LocalTime.now().toString()
→ 14:30:00
```

WITH TIMEZONE OFFSET

```
OffsetDateTime.now().toString()
→ 2024-12-02T14:30:00+01:00
```

WEEK DATE

```
LocalDate.now().format(DateTimeFormatter.ISO_WEEK_DATE)
→ 2024-W49-1
```

DURATION

```
Duration.ofHours(2).toString()
→ PT2H
```

PERIOD

```
Period.of(1, 2, 3).toString()
→ P1Y2M3D
```

PARSE ISO STRING

```
Instant.parse('2024-12-02T14:30:00Z')
```

C#

ROUND-TRIP FORMAT (FULL)

```
DateTime.UtcNow.ToString("o")
→ 2024-12-02T14:30:00.0000000Z
```

SORTABLE FORMAT

```
DateTime.Now.ToString("s")
→ 2024-12-02T14:30:00
```

DATE ONLY

```
DateTime.Now.ToString("yyyy-MM-dd")
→ 2024-12-02
```

UNIVERSAL SORTABLE

```
DateTime.UtcNow.ToString("u")
→ 2024-12-02 14:30:00Z
```

WITH OFFSET

```
DateTimeOffset.Now.ToString("o")
→ 2024-12-02T14:30:00.0000000+01:00
```

DURATION (TIMESPAN)

```
XmlConvert.ToString(TimeSpan.FromHours(2))
→ PT2H
```

PARSE ISO STRING

```
DateTime.Parse("2024-12-02T14:30:00Z")
```

GO

RFC3339 (ISO 8601)

```
time.Now().UTC().Format(time.RFC3339)
→ 2024-12-02T14:30:00Z
```

WITH NANOSECONDS

```
time.Now().Format(time.RFC3339Nano)
→ 2024-12-02T14:30:00.000000000Z
```

DATE ONLY

```
time.Now().Format("2006-01-02")
→ 2024-12-02
```

RUST

RFC3339 (CHRONO)

```
Utc::now().to_rfc3339()
→ 2024-12-02T14:30:00+00:00
```

DATE ONLY

```
Utc::now().format("%Y-%m-%d").to_string()
→ 2024-12-02
```

CUSTOM FORMAT

```
Utc::now().format("%Y-%m-%dT%H:%M:%S%.3fZ")
→ 2024-12-02T14:30:00.000Z
```

TIME ONLY

```
time.Now().Format("15:04:05")
```

→ 14:30:00

CUSTOM ISO FORMAT

```
time.Now().Format("2006-01-02T15:04:05-07:00")
```

→ 2024-12-02T14:30:00+01:00

PARSE ISO STRING

```
time.Parse(time.RFC3339, "2024-12-02T14:30:00Z")
```

WITH OPTIONS

```
Utc::now().to_rfc3339_opts(SecondsFormat::Secs, true)
```

→ 2024-12-02T14:30:00Z

DURATION (ISO8601 CRATE)

```
iso8601::duration::Duration::parse("PT2H30M")
```

PARSE ISO STRING

```
DateTime::parse_from_rfc3339("2024-12-02T14:30:00Z")
```

RUBY

FULL ISO DATETIME

```
Time.now.utc.iso8601
```

→ 2024-12-02T14:30:00Z

WITH FRACTIONAL SECONDS

```
Time.now.iso8601(3)
```

→ 2024-12-02T14:30:00.000+01:00

DATE ONLY

```
Date.today.iso8601
```

→ 2024-12-02

XMLSCHEMA FORMAT

```
Time.now.xmlschema
```

→ 2024-12-02T14:30:00+01:00

CUSTOM FORMAT

```
Time.now.strftime("%Y-%m-%dT%H:%M:%S%:z")
```

→ 2024-12-02T14:30:00+01:00

DURATION (ACTIVESUPPORT)

```
2.hours.iso8601
```

→ PT2H

PARSE ISO STRING

```
Time.iso8601("2024-12-02T14:30:00Z")
```

PHP

ISO 8601 FULL

```
date('c')
```

→ 2024-12-02T14:30:00+01:00

ATOM FORMAT

```
date(DATE_ATOM)
```

→ 2024-12-02T14:30:00+01:00

DATE ONLY

```
date('Y-m-d')
```

→ 2024-12-02

UTC WITH Z

```
gmdate('Y-m-d\TH:i:s\Z')
```

→ 2024-12-02T14:30:00Z

WEEK NUMBER

```
date('o-\WW-N')
```

→ 2024-W49-1

DATETIME OBJECT

```
(new DateTime())->format(DateTime::ATOM)
```

DURATION (DATEINTERVAL)

```
(new DateInterval('PT2H30M'))->format('PT%hH%iM')
```

→ PT2H30M

PARSE ISO STRING

```
new DateTime('2024-12-02T14:30:00Z')
```

SWIFT

ISO 8601 FULL

```
ISO8601DateFormatter().string(from: Date())
```

→ 2024-12-02T14:30:00Z

WITH FRACTIONAL SECONDS

```
fmt.formatOptions = [.withInternetDateTime, .withFractionalSeconds]
```

→ 2024-12-02T14:30:00.000Z

DATE ONLY

```
fmt.formatOptions = [.withFullDate]
```

→ 2024-12-02

TIME ONLY

```
fmt.formatOptions = [.withTime]
```

→ 14:30:00

CUSTOM DATEFORMATTER

```
df.dateFormat = "yyyy-MM-dd'T'HH:mm:ssZZZZZ"
```

DURATION (MEASUREMENT)

```
Measurement(value: 2, unit: UnitDuration.hours)
```

PARSE ISO STRING

```
ISO8601DateFormatter().date(from: "2024-12-02T14:30:00Z")
```

KOTLIN

ISO INSTANT

```
Instant.now().toString()
```

→ 2024-12-02T14:30:00Z

LOCALDATETIME

```
LocalDateTime.now().toString()
```

→ 2024-12-02T14:30:00

DATE ONLY

```
LocalDate.now().toString()
```

→ 2024-12-02

WITH TIMEZONE

```
ZonedDateTime.now().toString()
```

→ 2024-12-02T14:30:00+01:00[Europe/Paris]

WEEK DATE

```
LocalDate.now().format(DateTimeFormatter.ISO_WEEK_DATE)
```

→ 2024-W49-1

DURATION

```
Duration.ofHours(2).toString()
```

→ PT2H

PARSE ISO STRING

```
Instant.parse("2024-12-02T14:30:00Z")
```

C/C++

ISO 8601 DATETIME

```
strftime(buf, sizeof(buf), "%Y-%m-%dT%H:%M:%SZ", gmtime(&t))
```

→ 2024-12-02T14:30:00Z

DATE ONLY

```
strftime(buf, sizeof(buf), "%Y-%m-%d", localtime(&t))
```

→ 2024-12-02

TIME ONLY

```
strftime(buf, sizeof(buf), "%H:%M:%S", localtime(&t))
```

→ 14:30:00

WITH TIMEZONE OFFSET

```
strftime(buf, sizeof(buf), "%Y-%m-%dT%H:%M:%S%z", localtime(&t))
```

→ 2024-12-02T14:30:00+0100

WEEK NUMBER

```
strftime(buf, sizeof(buf), "%G-W%V-%u", localtime(&t))
```

DART

ISO 8601 FULL

```
DateTime.now().toUtc().toIso8601String()
```

→ 2024-12-02T14:30:00.000Z

LOCAL TIME

```
DateTime.now().toIso8601String()
```

→ 2024-12-02T14:30:00.000

DATE ONLY

```
DateFormat('yyyy-MM-dd').format(DateTime.now())
```

→ 2024-12-02

CUSTOM FORMAT

```
DateFormat("yyyy-MM-dd'T'HH:mm:ss").format(dt)
```

→ 2024-12-02T14:30:00

DURATION

```
Duration(hours: 2, minutes: 30).toString()
```

→ 2:30:00.000000

PARSE ISO STRING

```
DateTime.parse("2024-12-02T14:30:00Z")
```

→ 2024-W49-1

C++20 CHRONO

```
std::format("{:%FT%TZ}", std::chrono::system_clock::now())
```

PARSE (STRPTIME)

```
strptime("2024-12-02T14:30:00", "%Y-%m-%dT%H:%M:%S", &tm)
```

ELIXIR

ISO 8601 (UTC)

```
DateTime.utc_now() |> DateTime.to_iso8601()
```

→ 2024-12-02T14:30:00Z

WITH MICROSECONDS

```
DateTime.to_iso8601(dt, :extended)
```

→ 2024-12-02T14:30:00.000000Z

DATE ONLY

```
Date.utc_today() |> Date.to_iso8601()
```

→ 2024-12-02

TIME ONLY

```
Time.utc_now() |> Time.to_iso8601()
```

→ 14:30:00

NAIVEDATETIME

```
NaiveDateTime.utc_now() |> NaiveDateTime.to_iso8601()
```

→ 2024-12-02T14:30:00

DURATION (TIMEX)

```
Timex.Duration.to_string(Timex.Duration.from_hours(2))
```

→ PT2H

PARSE ISO STRING

```
DateTime.from_iso8601("2024-12-02T14:30:00Z")
```

R

ISO 8601 FORMAT

```
format(Sys.time(), "%Y-%m-%dT%H:%M:%SZ", tz="UTC")
```

→ 2024-12-02T14:30:00Z

DATE ONLY

```
format(Sys.Date(), "%Y-%m-%d")
```

→ 2024-12-02

WITH TIMEZONE

```
format(Sys.time(), "%Y-%m-%dT%H:%M:%S%z")
```

→ 2024-12-02T14:30:00+0100

WEEK NUMBER

```
format(Sys.Date(), "%G-W%V-%u")
```

→ 2024-W49-1

LUBRIDATE PACKAGE

```
lubridate::format_ISO8601(now())
```

→ 2024-12-02T14:30:00Z

PARSE ISO STRING

```
as.POSIXct("2024-12-02T14:30:00Z", format="%Y-%m-%dT%H:%M:%SZ")
```

PERL

ISO 8601 FORMAT

```
strftime("%Y-%m-%dT%H:%M:%SZ", gmtime())
```

→ 2024-12-02T14:30:00Z

DATE ONLY

```
strftime("%Y-%m-%d", localtime())
```

→ 2024-12-02

WITH TIMEZONE

```
strftime("%Y-%m-%dT%H:%M:%S%z", localtime())
```

→ 2024-12-02T14:30:00+0100

SCALA

ISO INSTANT

```
Instant.now.toString
```

→ 2024-12-02T14:30:00Z

LOCALDATETIME

```
LocalDateTime.now.toString
```

→ 2024-12-02T14:30:00

DATE ONLY

```
LocalDate.now.toString
```

→ 2024-12-02

DATETIME MODULE

```
DateTime->now->iso8601()
```

→ 2024-12-02T14:30:00

DATETIME WITH TZ

```
DateTime->now(time_zone => 'UTC')->strftime('%FT%TZ')
```

→ 2024-12-02T14:30:00Z

DURATION

```
DateTime::Duration->new(hours => 2)->iso8601()
```

→ PT2H

PARSE ISO STRING

```
Time::Piece->strptime($str, "%Y-%m-%dT%H:%M:%S")
```

ZONEDDATETIME

```
ZonedDateTime.now.toString
```

→ 2024-12-02T14:30:00+01:00[Europe/Paris]

CUSTOM FORMAT

```
LocalDateTime.now.format(DateTimeFormatter.ISO_LOCAL_DATE_TIME)
```

DURATION

```
Duration.ofHours(2).toString
```

→ PT2H

PARSE ISO STRING

```
Instant.parse("2024-12-02T14:30:00Z")
```

HASKELL

ISO 8601 FORMAT

```
formatTime defaultTimeLocale "%Y-%m-%dT%H:%M:%SZ" t
```

→ 2024-12-02T14:30:00Z

DATE ONLY

```
formatTime defaultTimeLocale "%Y-%m-%d" day
```

→ 2024-12-02

WITH TIMEZONE

```
formatTime defaultTimeLocale "%Y-%m-%dT%H:%M:%S%z" zt
```

→ 2024-12-02T14:30:00+0100

ISO8601 PACKAGE

```
iso8601Show utcTime
```

→ 2024-12-02T14:30:00Z

WEEK DATE

```
formatTime defaultTimeLocale "%G-W%V-%u" day
```

→ 2024-W49-1

PARSE ISO STRING

```
parseTimeM True defaultTimeLocale "%Y-%m-%dT%H:%M:%SZ" str
```

▸ RFC 3339 VS ISO 8601

Most languages produce RFC 3339 output, which is a strict profile of ISO 8601 designed for internet timestamps. RFC 3339 requires a full date-time with timezone, while ISO 8601 allows many more variations.

▸ DURATION SUPPORT

Native ISO 8601 duration support varies by language. Java, Ruby, and .NET have built-in support. For others, consider libraries like isodate (Python),iso8601 (Rust), or Timex (Elixir).

ISO8601.COM
TECHNICAL REFERENCE

Comprehensive documentation of the international standard for date and time representation.

QUICK REFERENCE

Date	YYYY-MM-DD
Time	hh:mm:ss
UTC	...Z
Combined	...T...
Duration	P...T...

OFFICIAL SOURCES

ISO Official Page →
RFC 3339 →
Wikipedia Article →

This site is an unofficial educational resource. ISO 8601 is a trademark of the International Organization for Standardization.

LAST UPDATED: 2024-12-02T00:00:00Z

END OF DOCUMENT — ISO8601.COM — MACHINE-READABLE DATE FORMAT STANDARD