Date and time with Luxon

Luxon is a JavaScript library that makes it easier to work with date and time. For full details of how to use Luxon, refer to Luxon's documentation.

n8n passes dates between nodes as strings, so you need to parse them. Luxon makes this easier.

Python support

Luxon is a JavaScript library. The two convenience variables created by n8n are available when using Python in the Code node, but their functionality is limited:

- You can't perform Luxon operations on these variables. For example, there is no Python equivalent for `$today.minus(...)`.
- The generic Luxon functionality, such as Convert date string to Luxon, isn't available for Python users.

Variables

n8n uses Luxon to provide two custom variables:

- `now`: a Luxon object containing the current timestamp. Equivalent to `DateTime.now()`.
- `today`: a Luxon object containing the current timestamp, rounded down to the day. Equivalent to `DateTime.now().set({ hour: 0, minute: 0, second: 0, millisecond: 0 })`.

Note that these variables can return different time formats when cast as a string. This is the same behavior as Luxon's `DateTime.now()`.

Expressions (JavaScript)

```
{{$now}}
// n8n displays the ISO formatted timestamp
// For example 2022-03-09T14:02:37.065+00:00
{{"Today's date is " + $now}}
// n8n displays "Today's date is <unix timestamp>"
// For example "Today's date is 1646834498755"
```

Code node (JavaScript)

```
$now
// n8n displays <ISO formatted timestamp>
// For example 2022-03-09T14:00:25.058+00:00
let rightNow = "Today's date is " + $now
// n8n displays "Today's date is <unix timestamp>"
// For example "Today's date is 1646834498755"
```

Code node (Python)

```
_now
# n8n displays <ISO formatted timestamp>
# For example 2022-03-09T14:00:25.058+00:00
rightNow = "Today's date is " + str(_now)
# n8n displays "Today's date is <unix timestamp>"
# For example "Today's date is 1646834498755"
```

n8n provides built-in convenience functions to support data transformation in expressions for dates. Refer to Data transformation functions | Dates for more information.

Date and time behavior in n8n

Be aware of the following:

- In a workflow, n8n converts dates and times to strings between nodes. Keep this in mind when doing arithmetic on dates and times from other nodes.
- With vanilla JavaScript, you can convert a string to a date with `new Date('2019-06-23')`. In Luxon, you must use a function explicitly stating the format, such as `DateTime.fromISO('2019-06-23')` or `DateTime.fromFormat("23-06-2019", "dd-MM-yyyy")`.

Setting the timezone in n8n

Luxon uses the n8n timezone. This value is either:

- Default: `America/New York`
- A custom timezone for your n8n instance, set using the `GENERIC_TIMEZONE` environment variable.
- A custom timezone for an individual workflow, configured in workflow settings.

Common tasks

This section provides examples for some common operations. More examples, and detailed guidance, are available in Luxon's own documentation.

Convert date string to Luxon

You can convert date strings and other date formats to a Luxon DateTime object. You can convert from standard formats and from arbitrary strings.

A difference between Luxon DateTime and JavaScript Date

With vanilla JavaScript, you can convert a string to a date with `new Date('2019-06-23')`. In Luxon, you must use a function explicitly stating the format, such as `DateTime.fromISO('2019-06-23')` or `DateTime.fromFormat("23-06-2019", "dd-MM-yyyy")`.

If you have a date in a supported standard technical format:

Most dates use `fromISO()`. This creates a Luxon DateTime from an ISO 8601 string. For example:

Expressions (JavaScript)

```
{{DateTime.fromISO('2019-06-23T00:00:00.00')}}
```

Code node (JavaScript)

```
let luxonDateTime = DateTime.fromISO('2019-06-23T00:00:00.00')
```

Luxon's API documentation has more information on fromISO.

Luxon provides functions to handle conversions for a range of formats. Refer to Luxon's guide to Parsing technical formats for details.

If you have a date as a string that doesn't use a standard format:

Use Luxon's Ad-hoc parsing. To do this, use the `fromFormat()` function, providing the string and a set of tokens that describe the format.

For example, you have n8n's founding date, 23rd June 2019, formatted as `23-06-2019`. You want to turn this into a Luxon object:

Expressions (JavaScript)

```
{{DateTime.fromFormat("23-06-2019", "dd-MM-yyyy")}}
```

Code node (JavaScript)

```
let newFormat = DateTime.fromFormat("23-06-2019", "dd-MM-yyyy")
```

When using ad-hoc parsing, note Luxon's warning about Limitations. If you see unexpected results, try their Debugging guide.

Get n days from today

Get a number of days before or after today.

Expressions (JavaScript)

For example, you want to set a field to always show the date seven days before the current date.

In the expressions editor, enter:

```
{{$today.minus({days: 7})}}
```

On the 23rd June 2019, this returns `[Object: "2019-06-16T00:00:00.000+00:00"]`.

This example uses n8n's custom variable `$today` for convenience. It's the equivalent of `DateTime.now().set({ hour: 0, minute: 0, second: 0, millisecond: 0 }).minus({days: 7})`.

Code node (JavaScript)

For example, you want a variable containing the date seven days before the current date.

In the code editor, enter:

```
let sevenDaysAgo = $today.minus({days: 7})
```

On the 23rd June 2019, this returns `[Object: "2019-06-16T00:00:00.000+00:00"]`.

This example uses n8n's custom variable `$today` for convenience. It's the equivalent of `DateTime.now().set({ hour: 0, minute: 0, second: 0, millisecond: 0 }).minus({days: 7})`.

For more detailed information and examples, refer to:

- Luxon's guide to math
- Their API documentation on DateTime plus and DateTime minus

Create human-readable dates

In Get n days from today, the example gets the date seven days before the current date, and returns it as `[Object: "yyyy-mm-dd-T00:00:00.000+00:00"]` (for expressions) or `yyyy-mm-dd-T00:00:00.000+00:00` (in the Code node). To make this more readable, you can use Luxon's formatting functions.

For example, you want the field containing the date to be formatted as DD/MM/YYYY, so that on the 23rd June 2019, it returns `23/06/2019`.

This expression gets the date seven days before today, and converts it to the DD/MM/YYYY format.

Expressions (JavaScript)

```
{{$today.minus({days: 7}).toLocaleString()}}
```

Code node (JavaScript)

```
let readableSevenDaysAgo = $today.minus({days: 7}).toLocaleString()
```

You can alter the format. For example:

Expressions (JavaScript)

```
{{$today.minus({days: 7}).toLocaleString({month: 'long', day: 'numeric', year: 'numeric'})}}
```

On 23rd June 2019, this returns "16 June 2019".

Code node (JavaScript)

```
let readableSevenDaysAgo = $today.minus({days: 7}).toLocaleString({month: 'long', day: 'numeric', year: 'numeric'})
```

On 23rd June 2019, this returns "16 June 2019".

Refer to Luxon's guide on toLocaleString (strings for humans) for more information.

Get the time between two dates

To get the time between two dates, use Luxon's diffs feature. This subtracts one date from another and returns a duration.

For example, get the number of months between two dates:

Expressions (JavaScript)

```
{{DateTime.fromISO('2019-06-23').diff(DateTime.fromISO('2019-05-23'),
```

```
    'months').toObject()}}
```

This returns `[Object: {"months":1}]`.

Code node (JavaScript)

```
let monthsBetweenDates = DateTime.fromISO('2019-06-23').diff(DateTime.fromISO('2019-05-23'), 'months').toObject()
```

This returns `{"months":1}`.

Refer to Luxon's Diffs for more information.

A longer example: How many days to Christmas?

This example brings together several Luxon features, uses JMESPath, and does some basic string manipulation.

The scenario: you want a countdown to 25th December. Every day, it should tell you the number of days remaining to Christmas. You don't want to update it for next year - it needs to seamlessly work for every year.

Expressions (JavaScript)

```
{{"There are " + $today.diff(DateTime.fromISO($today.year + '-12-25'), 'days').toObject().days.toString().substring(1) + " days to Christmas!"}}
```

This outputs `"There are <number of days> days to Christmas!"`. For example, on 9th March, it outputs "There are 291 days to Christmas!".

A detailed explanation of what the expression does:

- `{{`: indicates the start of the expression.

- `"There are "`: a string.
- `+`: used to join two strings.
- `$today.diff()`: This is similar to the example in Get the time between two dates, but it uses n8n's custom `$today` variable.
- `DateTime.fromISO($today.year + '-12-25'), 'days'`: this part gets the current year using `$today.year`, turns it into an ISO string along with the month and date, and then takes the whole ISO string and converts it to a Luxon DateTime data structure. It also tells Luxon that you want the duration in days.
- `toObject()` turns the result of diff() into a more usable object. At this point, the expression returns `[Object: {"days":-<number-of-days>}]`. For example, on 9th March, `[Object: {"days":-291}]`.
- `.days` uses JMESPath syntax to retrieve just the number of days from the object. For more information on using JMESPath with n8n, refer to our JMESpath documentation. This gives you the number of days to Christmas, as a negative number.
- `.toString().substring(1)` turns the number into a string and removes the `-`.
- `+ " days to Christmas!"`: another string, with a `+` to join it to the previous string.
- `}}`: indicates the end of the expression.

Code node (JavaScript)

```
let daysToChristmas = "There are " +
$today.diff(DateTime.fromISO($today.year + '-12-25'),
'days').toObject().days.toString().substring(1) + " days to
Christmas!";
```

This outputs `"There are <number of days> days to Christmas!"`. For example, on 9th March, it outputs "There are 291 days to Christmas!".

A detailed explanation of what the code does:

- `"There are "`: a string.
- `+`: used to join two strings.
- `$today.diff()`: This is similar to the example in Get the time between two dates, but it uses n8n's custom `$today` variable.
- `DateTime.fromISO($today.year + '-12-25'), 'days'`: this part gets the current year using `$today.year`, turns it into an ISO string along with the month and date, and then takes the whole ISO string and converts it to a Luxon DateTime data structure. It also tells Luxon that you want the duration in days.
- `toObject()` turns the result of diff() into a more usable object. At this point, the expression returns `[Object: {"days":-<number-of-days>}]`. For example, on 9th March, `[Object: {"days":-291}]`.
- `.days` uses JMESPath syntax to retrieve just the number of days from the object. For more information on using JMESPath with n8n, refer to our JMESpath documentation. This gives you the number of days to Christmas, as a negative number.
- `.toString().substring(1)` turns the number into a string and removes the `-`.
- `+ " days to Christmas!"`: another string, with a `+` to join it to the previous string.