# Handling a One-Time Exception on an Otherwise Regular Schedule

> A weekly meeting that skips one specific week for a holiday is still "weekly," and the exception needs to be layered on top of the rule rather than rewritten as a new rule.

A recurring schedule frequently needs a one-off exception, whether that's a specific occurrence canceled, moved to a different date, or replaced entirely, without changing the underlying recurrence rule for every other occurrence. Modeling this as a base rule plus a small list of dated exceptions (in calendaring terms, an EXDATE for a removed occurrence and an RDATE for an added replacement) keeps the schedule's regular pattern intact and auditable, rather than requiring the entire rule to be redefined around one irregular week.

## Weekly rule, four occurrences, with one holiday exception




A base recurrence rule stays unchanged; a single occurrence is removed (and optionally replaced by a moved date) as an explicit exception layered on top.

### Why an exception should not be modeled as a rule change

Redefining the entire recurrence rule to work around one irregular week (splitting it into "weekly until week 2, then weekly again starting week 3") loses the fact that the schedule is still fundamentally one weekly series with a single documented exception, and makes every future exception require another rule split. Tracking exceptions as a small explicit list against an unchanged base rule keeps the schedule's regular pattern visible and each exception individually auditable.

### Two kinds of exception: removal and replacement

A pure cancellation removes one occurrence with nothing put in its place. A moved occurrence removes the original date and adds a replacement date elsewhere, which is really two linked

exceptions (a removal and an addition) rather than a single operation, since the replacement date does not have to follow the base rule's own pattern at all.

## What the Schedule Exception Resolver does

The Schedule Exception Resolver takes a base recurrence rule (built the same way as the Recurring Date Generator's rule) plus an explicit list of removed and replacement dates, and produces the final resolved occurrence list with every exception applied, so the difference between "what the base rule alone would generate" and "what the schedule actually is after known exceptions" is visible side by side.

> **Key takeaway.** Model a one-off schedule exception as a removal (and optional replacement) layered on an unchanged base recurrence rule, not as a redefinition of the rule itself. This keeps the regular pattern intact and each exception individually visible.

## Frequently asked

### Does this affect the Anniversary Planner's observed-date handling for a leap-day anniversary?

That is a related but distinct mechanism. An observed-date rule for a leap-day anniversary is a standing, repeating substitution built into the rule itself, whereas the exceptions covered here are one-off, individually dated overrides on top of an otherwise unmodified rule.

### Can a recurrence have more exceptions than actual occurrences?

In principle a heavily exception-laden schedule can end up looking very different from its base rule, at which point it may be worth asking whether the base rule is still the right model at all, rather than a sign of any error in the exception-handling logic itself.

### Do exceptions need to be entered in date order?

No, the Schedule Exception Resolver matches each exception against the base rule's generated occurrences regardless of the order exceptions are entered in, and reports plainly if an exception date does not match any generated occurrence at all.

WhenMath is a browser-local calculation tool, not a legal, payroll, or scheduling authority. This guide explains a calculation convention; it is educational, not a determination that any specific contract, employer, or jurisdiction follows it.