Moved to seven, with nothing to send it

Tara’s ritual scheduler moves an evening invitation past quiet hours, and no code was found that sends its reminders.

TARA · CONTINUE · CHAPTER 3 OF 5

Three things this chapter teaches, carried by one Monday evening.

RUNNING EXAMPLE

Monday 21:50 UTC, a recovery window, a preferred time of 22:15 and quiet hours from 22:00 to 07:00: the invitation moves to Tuesday 07:00, the primary reminder shifts from 06:45 to 07:00, a follow-up lands at 08:30 — and no code was found that sends any of them.

LISTENING TIME

About 27 min · 20 slides

YOU WILL LEARN

1. **The invitation moves**
   Preferred time, continuity, then quiet hours, in that order.
2. **Reminders shift too**
   Each gets its own quiet-hours pass; a flag records the move.
3. **Records, not messages**
   Saved in memory, returned by a query, sent by no caller found.

By the end you can say where an invitation lands, why each reminder lands where it does, and what would have to send them.

Where the invitation lands

The evening case, the plan it produces, the policy around it, and the two rules that pick a time.

Section 1 of 3

1 Where the invitation lands
Preferred time, continuity and a default clock.

2 Zones, quiet hours, reminders
Wall clocks, a quiet window, a second shift.

3 Records nobody sends
Two clocks, a memory store, a due query.

The invitation’s time comes from a preference clock and a supplied continuity state.

Monday 21:50, and a recovery window

The member is in a recovery window; the scheduler plans a gentle return, and quiet hours move it.

Illustrative case · executed library behaviour · pinned clock

22:15 becomes 07:00 tomorrow

Monday 21:50 UTC; quiet hours 22:00 to 07:00, every day.

Preferred time · 22:15

Still ahead of 21:50, so the recovery rule keeps it.

Quiet hours · Tuesday 07:00

22:15 is inside the window; the end of it is 07:00.

Reminders · 07:00 and 08:30

The primary shifts from 06:45; the follow-up stays.

The pinned scheduler, run with declared inputs; no reminder is sent.

> Read the final time and its reason before promising a reminder.

A plan: one invitation, its reminders

Identity, the resolved time, continuity and reasoning travel in one returned object.

Executed plan · declared inputs · pinned clock

TaraRitualSchedulePlan · the evening case

planId · userId · templateId	`tara-ritual-plan-… · evening-return`	Names the subject, template and moment.
scheduledAt · timezone	`2026-09-08T07:00Z · UTC`	An absolute instant, after quiet hours.
continuity · status	`recovery_window · scheduled`	Supplied continuity; status from the supplied now.
reminders · reasoning	`2 reminders · 2 lines`	Separate records, and a same-day sentence.

One returned plan, read field by field.

> Keep the plan’s state separate from each reminder’s state.

Where the schedule rules are written

The contract, the trigger model and the plan builder each describe part of the problem.

CONTRACT RULES

Cadence, conflicts, misses

Quiet hours respected; skip, defer or ask; one recovery.

BUILDER READS

none of these

DOMAIN TRIGGER

Recurrence and admission

Priority, enabled, window, minimum hours since completion.

BUILDER READS

the first daypart only

PLAN BUILDER

A clock and a state

Preferred time, supplied continuity, quiet hours.

BUILDER READS

all three

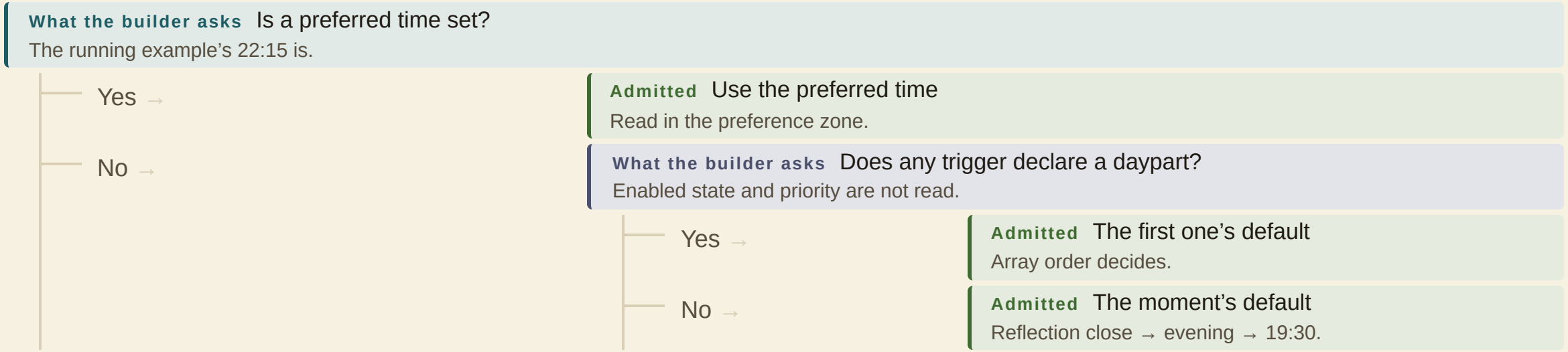
Declared policy against the fields the builder actually reads.

> Trace a policy field into the code that enforces it.

No preferred time, then the first daypart

An explicit preferred time wins; otherwise the builder derives a default from the template.

A template and the member’s preferences arrive at the builder.



How resolveScheduledTime chooses the wall clock before continuity applies.

> The daypart fallback reads declaration order; it does not rank or admit triggers.

Where the member is moves the candidate

The caller supplies the continuity state, and each state picks the candidate its own way.

Supplied continuity	Candidate before quiet hours
resume_window · after_interruption	the later of now and last activity, plus 15 minutes
missed_window · recovery_window	today’s preferred time if strictly later; otherwise now plus 30 minutes
recent_completion	tomorrow at the preferred time
fresh_start and anything else	today’s preferred time if still ahead; otherwise tomorrow’s

> Compute the candidate first, then look at every later adjustment.

Zones, quiet hours, reminders

How a wall clock becomes an instant, how quiet hours move it, and how each reminder is moved again.

Section 2 of 3

1 Where the invitation lands
Preferred time, continuity and a default clock.

2 Zones, quiet hours, reminders
Wall clocks, a quiet window, a second shift.

3 Records nobody sends
Two clocks, a memory store, a due query.

Every time the scheduler produces passes through quiet hours on its own.

One wall clock, zero, one or two instants

Repeated times resolve to the earlier instant and skipped times move forward.

A NORMAL CLOCK

One instant

UTC · 7 September 2026 · 07:30.

RETURNED

07:30Z · unique

CLOCKS GO BACK

Two instants, the earlier

New York · 1 November 2026 · 01:30.

RETURNED

05:30Z, before 06:30Z

CLOCKS GO FORWARD

No instant, shifted

New York · 8 March 2026 · 02:30.

RETURNED

07:30Z · 03:30 local

resolveTaraZonedDateTime, executed with time-zone data 2026a.

> Keep the resolution reason if a member needs to know why a time moved.

Quiet hours move a time to their end

The quiet window has its own time zone and weekdays; start is in, end is out.

A candidate instant arrives with the member’s quiet-hours settings.

What the step asks Enabled, on an active day?
The weekday is read in the quiet-hours zone.

- No → **Admitted** Returned unchanged
  Disabled, or not an active day.
- Yes → **What the step asks** Inside the window?
  Start included, end excluded.
  - No → **Admitted** Returned unchanged
    Outside the window.
  - Yes → **Try again** Moved to the window’s end
    Monday 22:15 → Tuesday 07:00.

shiftForQuietHours, applied to every instant the scheduler produces.

Check the candidate in the quiet-hours zone, at both edges.

A Monday night that stops at midnight

An overnight window tests the candidate’s own weekday, not the night it began on.

MONDAY ONLY

Monday 22:15

Monday is active and the time is quiet.

RETURNED

moved to Tuesday 07:00

MONDAY ONLY

Tuesday 06:00

Tuesday is not an active day.

RETURNED

left at Tuesday 06:00

EVERY DAY

Tuesday 06:00

Tuesday is active and the time is quiet.

RETURNED

moved to Tuesday 07:00

Illustrative weekday settings; the helper tests the candidate’s own day.

> Explain active days with a case that crosses midnight.

The reminder is moved a second time

A lead time can collapse onto the invitation when quiet hours apply twice.

Illustrative case · executed library behaviour · pinned clock

07:00 − 15 min → 06:45 → 07:00

The primary reminder rejoins the invitation at the end of quiet hours.

The invitation · 07:00

Already moved by quiet hours.

The plain lead · 06:45

Fifteen minutes before, and quiet too.

The reminder · 07:00, flagged

Moved again; still in the list.

The primary is moved, not removed; the follow-up stays at 08:30.

> The quiet-hours flag records a move; the reminder is still there.

Reminders off leaves the invitation

When reminders are on, only a miss or recovery earns the follow-up.

REMINDERS OFF

An empty list

enabled false; the invitation still returns.

RETURNS

plan kept · no reminders

REMINDERS ON

A primary reminder

Wording follows the continuity state.

RETURNS

always one

RECOVERY REMINDERS ON

One follow-up

Only after a miss or in recovery.

RETURNS

08:30 in the evening case

buildReminderPlans; building records sends nothing.

> Read the returned reminder list; do not infer it from the plan.

Records nobody sends

Two clocks behind two statuses, a store in memory, two queries, a reschedule, and the send that is missing.

Section 3 of 3

1 Where the invitation lands
Preferred time, continuity and a default clock.

2 Zones, quiet hours, reminders
Wall clocks, a quiet window, a second shift.

3 Records nobody sends
Two clocks, a memory store, a due query.

> A saved, due reminder is a record; sending it is somebody else’s job.

Two statuses, two clocks

The plan compares with the supplied now; each reminder compares with the runtime clock.

	Plan status Set by the plan builder.	Reminder status Set by the reminder builder.
Compared with	the supplied now	the runtime clock
Illustration	06:00 now, invitation 07:30	08:00 clock, reminder 07:15
Result	scheduled	due
The evening case	scheduled	scheduled

> The plan can read scheduled while its reminder reads due; neither says anything was sent.

Status comparisons in two functions, against two clocks.

> Pin both clocks when reproducing a status.

Saved in a map, in memory

The default store clones plans in and out, and lives as long as the process.

Pinned implementation · executed by the harness

The default store · in memory

savePlan	`planId → structuredClone(plan)`	Clones the incoming plan into the map.
loadPlan	`a clone, or null`	Returns a copy, never the stored object.
loadAllPlans	`a clone of every plan`	What both queries start from.
deletePlan	`removes the key`	Removes a key; saving the same id replaces a plan.

Process-local storage operations; nothing here survives the process.

> A plan saved in memory is not a scheduled notification.

Due by the clock, not ready to send

The two queries filter by time, and leave lifecycle and delivery to their caller.

	Upcoming plans getUpcomingPlans(member, now).	Due reminders getDueReminders(now).
Whose	the requested member	every member
When	invitation at or after now	reminder at or before now
Status checked	no	no
Marks as sent	nothing to mark	no
The evening case	the plan at Monday 22:00; none at 07:05	the primary at 07:05, twice

> Both answer when, not whether: neither checks status, and the due query does not even ask whose.

getUpcomingPlans and getDueReminders; both return records and change nothing.

> Check lifecycle and delivery state before acting on a returned reminder.

Rescheduling keeps one identifier, not all

The outer plan identifier is restored; the reminders inside keep the rebuilt one.

Executed by the harness · A and B stand for generated identifiers

Reschedule result · identity specimen

plan.planId	`plan A`	Restored after the rebuild.
reminders[].planId	`plan B`	Kept from the rebuild.
reminders[].reminderId	`plan B + primary · follow_up`	Built from the rebuilt identifier.

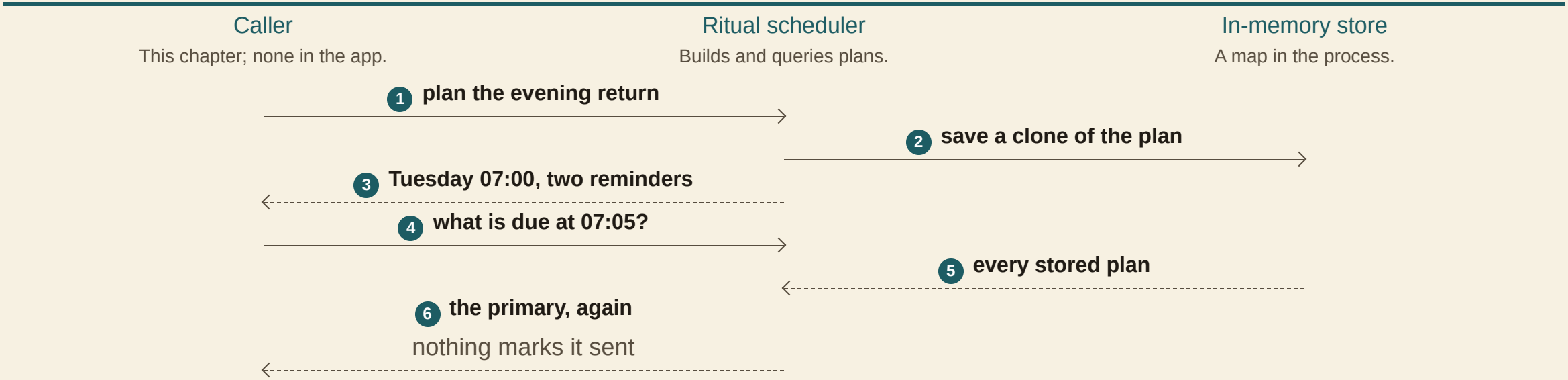
The helper returns this data; it saves and cancels nothing.

> Check parent and child identifiers before replacing a stored plan.

Due at seven, and nothing sends it

The evening case, run through the scheduler, ends as a due record and no message.

A caller plans the evening return, then asks what is due at 07:05 on Tuesday.



The evening case through TaraRitualScheduler and its default store.

> Confirm the time from the plan, and delivery only from a sender’s own evidence.

What to carry out of this chapter

Three habits for reading any schedule a library builds.

REMEMBER

1. **Read the final time**
   The reasoning says same day; the timestamp says tomorrow at seven.
2. **A flag is not a removal**
   A moved reminder stays in the list, marked, and the due query returns it.
3. **Due is not sent**
   The due query returns records; nothing marks, sends or cancels them.

NEXT · CHAPTER 4 OF 5

Tara consumer continuity: checkpoints, controls and saved outputs

Next: continuity — one interrupted sitting across four systems, and what is true of it at 2:20. →

> Every time in this chapter is exact, and none of the reminders reaches anyone.