Three records, one word: completed

A lifecycle contract, a domain evaluator and player telemetry each decide it, and none of them converts.

TARA · PRACTISE · CHAPTER 2 OF 5

Three things this chapter teaches, carried by one ritual with an unfinished passage.

RUNNING EXAMPLE

A three-step ritual — a four-minute guided practice, a three-minute required passage and a four-minute optional journal — with the practice and the journal complete and the passage skipped.

LISTENING TIME

About 41 min · 27 slides

YOU WILL LEARN

1. **Three records say completed**
   A lifecycle contract, a domain evaluator and player telemetry, with three vocabularies.
2. **The meter is not the decision**
   Under the default policy it can read a hundred per cent with required work unfinished.
3. **Recovery is a proposal**
   The shorter template it builds keeps the original duration minimum.

By the end you can name which record produced a completion, and what its percentage was measuring.

Three records, one word

A lifecycle contract, a domain evaluator and player telemetry, each deciding completed its own way.

Section 1 of 4

1 Three records, one word
A lifecycle contract, a domain evaluator and player telemetry, each deciding completed its own way.

2 What the session contract checks
An event with a consequence, a history that must agree with itself, and an audio floor no template lowers.

3 The evaluator, and the meter beside it
Required counts, a required ratio and an actual duration — and a percentage that answers none of them.

4 What recovery proposes, and what it proves
A shorter template with the original policy, a stable completion object, and a link that routes rather than syncs.

> Name the record before reading the word; no inspected code converts one into another.

A practice stops without losing its next step

The running example, and the three answers a useful return has to separate.

Illustrative case · executed evaluator input

“Can I return to the passage?”

The guided practice is done. A required passage remains. Journaling is optional.

Guided practice · 4 min

Required, and completed before the interruption.

Required passage · 3 min

Skippable only if its own step policy allows it.

Optional journal · 4 min

Its completion cannot satisfy a required-step count.

Durations and progress are illustrative input to the actual domain evaluator.

> Keep the unfinished work, the completion decision and the return evidence visible at once.

Three records, three completion vocabularies

The words overlap and the rules do not, so a claim has to name which record produced it.

LIFECYCLE CONTRACT

RitualSession

Template version, an event history, step progress and continuation.

ITS OWN WORDS FOR COMPLETED

completed · partially-completed

DOMAIN EVALUATOR

Execution record

Required work, a required ratio, an actual duration and an adaptation policy.

ITS OWN WORDS FOR COMPLETED

completed · completed_with_adjustments · ended_early

PLAYER TELEMETRY

SittingCompletionEvent

Duration, method, state and optional reflection, emitted per sitting.

ITS OWN WORDS FOR COMPLETED

its own timing guard, and several events per sitting

No inspected code converts any one of these three into another.

> Name the record before interpreting completed, partial or resume.

A session binds progress to a template version

Identity, bounded evidence arrays, state timestamps and continuation stay inspectable together.

Source field specimen · no member data

RitualSession · source fields

Identity and version	`id · userId · templateId · templateVersion`	Names this session and the exact template it was run against.
Practice evidence	`stepProgress · audioCompletionPercent`	Up to 24 step entries, plus a bounded audio percentage.
Lifecycle evidence	`state · events · completedAt · abandonedAt`	Up to 200 events, with the timestamps the state requires.
Continuation	`continuationState`	Program, teacher, theme, prerequisites and governance scope.

Grouped for reading; the schema neither loads nor executes the referenced template.

> Retain the version and the evidence needed to explain the state the record is in.

Every state-changing edge the contract allows

Two states are terminal, one looks terminal and is not, and paused cannot complete directly.

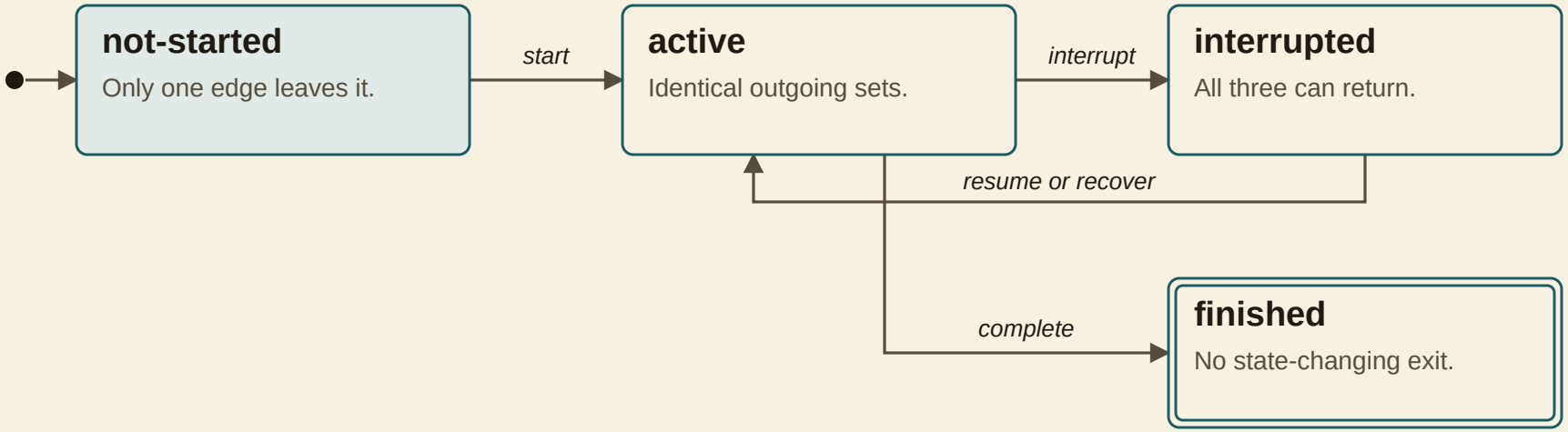
Current state	Allowed next states
not-started	started
started	paused · drifted · abandoned · completed · partially-completed
resumed	paused · drifted · abandoned · completed · partially-completed
paused	drifted · resumed · abandoned
drifted	resumed · abandoned
abandoned	resumed
completed	No state-changing exit
partially-completed	No state-changing exit

> Completed and partially-completed are terminal; abandoned can still resume.

The shape those edges make

Started and resumed behave identically, and three interruptible states all return to the same place.

Exact lifecycle relationships · no timer or interface assertion



The lifecycle shape. The exact eight-state edge list is the table beside it.

> A drift event records a lifecycle change, not a measured idle threshold.

What the session contract checks

An event with a consequence, a history that must agree with itself, and an audio floor no template lowers.

Section 2 of 4

1 Three records, one word
A lifecycle contract, a domain evaluator and player telemetry, each deciding completed its own way.

2 What the session contract checks
An event with a consequence, a history that must agree with itself, and an audio floor no template lowers.

3 The evaluator, and the meter beside it
Required counts, a required ratio and an actual duration — and a percentage that answers none of them.

4 What recovery proposes, and what it proves
A shorter template with the original policy, a stable completion object, and a link that routes rather than syncs.

> Every check here is exact, and every one of them is narrower than the sentence a reader would write about it.

Every action carries a reason and a consequence

A state label alone cannot explain an edit, a skip or a changed schedule.

Source field specimen · no member data

RitualSessionEvent · source fields

Who, and when	`eventId · type · timestamp · actor · reason`	A declared actor, a timestamp and a non-empty reason.
State before and after	`priorState → newState`	Adjacent events must agree on the handoff between them.
Work and scheduling	`stepId · scheduleChanged · changedFields`	Names the step, the changed fields and a progress figure.
Recorded consequence	`completed · partialCompletion · recoveryOpened`	Completion, recovery and an explanatory message the schema only stores.

The fields describe the action; the schema executes no scheduling, recovery or authorisation.

> Read who acted, what changed, and what the action meant for the practice.

A history has to agree with itself

Adjacency, legal edges, action semantics and the final state are checked together, and three things are not.

A supplied event array, read in array order from its first priorState.

Check Do the supplied events agree?

Adjacency, legal edges, action flags, and the session’s final state.

yes →

Admitted The history checks pass

Every other schema field still has to validate separately.

no →

Refused Reject the record

Issues are returned at the inconsistent fields, not on the array as a whole.

This gate does not establish complete history, timestamp order, authorisation or a save.

A consistent supplied history is not necessarily a complete or chronological one.

An eighty per cent floor no template lowers

The session schema rejects both completion states below it, and partial must also stay under a hundred.

	completed A terminal state with no state-changing exit.	partially-completed Also terminal, and bounded on both sides.
Audio percentage	At least 80, up to and including 100.	At least 80, and strictly below 100.
Required timestamp	completedAt must be present.	completedAt must be present.
Does the parser read the template?	No — it receives an identifier and a version.	No — the same identifier and version.
Does it derive this from the steps?	No — the step array is not consulted.	No — the step array is not consulted.

> Clearing the floor says nothing about whether the ritual’s required steps were performed.

Two terminal states, one shared floor, and one extra ceiling.

> Both completed states need at least eighty per cent; partial must also remain below a hundred.

Player telemetry has its own timing rule

A ratio guard that only runs when a planned duration is supplied, and a sixty-second ceiling on abandonment.

COMPLETED, POSITIVE PLAN

Duration at least 85% of plan

The only case where the ratio guard can refuse an event.

WHAT REMAINS UNPROVEN

that the ritual’s required steps were performed

COMPLETED, NULL PLAN

The check is skipped

A null planned duration removes the comparison entirely.

WHAT REMAINS UNPROVEN

that the planned duration was meaningful

COMPLETED, ZERO PLAN

Zero is admitted

Zero over zero is not-a-number, and the less-than test does not reject it.

WHAT REMAINS UNPROVEN

that the comparison was useful at all

ABANDONED

At most sixty seconds

Partially-completed carries no ratio floor in this refinement.

WHAT REMAINS UNPROVEN

the member’s reason for leaving

One refinement, four cases, and a floor that is absent from the partial state.

> An accepted player event does not establish that the ritual’s required steps were completed.

Continuation carries context, not permission

Program, teacher, theme, prerequisites and a governance scope travel in one record.

Source field specimen · no member data

ContinuationState · source fields

Program arc	`programArc · status`	Counts and the current index stay within the declared total.
Practice history	`teacherContinuity · themeContinuity`	Prior teacher and theme relationships, with a declared depth.
Next practice	`prerequisites · nextRecommendation`	A recommendation with a readiness claim and blocked identifiers.
Governance context	`memoryScope · consentRecordId`	A declared storage purpose and a consent reference nothing here checks.

The record carries context; an owning consumer has to enforce permission and readiness.

> The next recommendation needs an explicit readiness decision and an enforcing consumer.

A readiness claim, checked for consistency

The refinement compares the claim with the blocked list; it never recomputes the prerequisites.

CLAIM: NOT SATISFIED

Blocked IDs are checked

Non-empty, unique, and each one must name an unsatisfied entry.

WHAT THE BRANCH SETTLES

the listed identifiers are real and unsatisfied

CLAIM: SATISFIED

Only the list must be empty

No check that the prerequisites array agrees with the claim.

WHAT THE BRANCH SETTLES

nothing about whether the member is ready

STILL OWED

The applicable set, and the gate

Which prerequisites apply here, and who is permitted to proceed.

WHAT THE BRANCH SETTLES

the decision a launch surface has to make

Timestamp consistency is checked per prerequisite; the overall claim is never recomputed.

> A satisfied flag with an empty blocked list still needs an independent readiness decision.

The evaluator, and the meter beside it

Required counts, a required ratio and an actual duration — and a percentage that answers none of them.

Section 3 of 4

1 Three records, one word
A lifecycle contract, a domain evaluator and player telemetry, each deciding completed its own way.

2 What the session contract checks
An event with a consequence, a history that must agree with itself, and an audio floor no template lowers.

3 The evaluator, and the meter beside it
Required counts, a required ratio and an actual duration — and a percentage that answers none of them.

4 What recovery proposes, and what it proves
A shorter template with the original policy, a stable completion object, and a link that routes rather than syncs.

Read shouldCountAsCompleted for the decision; the percentage is answering a different question.

Three thresholds, and one permission

The count, the ratio and the duration must all pass, and a skipped requirement needs the template to allow it.

The running ritual: practice complete, journal complete, required passage skipped.

Test Was any step attempted?
Nothing attempted ends the evaluation immediately.

- no → **Refused** not_started
  No progress entry moved any step out of pending.
- yes → **Test** Do count, ratio and duration all hold?
  A skipped requirement also needs allowAdaptedCompletion on the policy.
  - all three hold → **Admitted** completed, or with adjustments
    Adjusted when a required step was skipped and adaptation was permitted.
  - any one fails → **Try again** in_progress, or ended_early
    Active or remaining non-terminal work keeps it open; an all-terminal set ends it.

Completion is evaluated before the active-step fallback; the percentage is not this decision.

> Adaptation permits a skipped requirement; it waives none of the three thresholds.

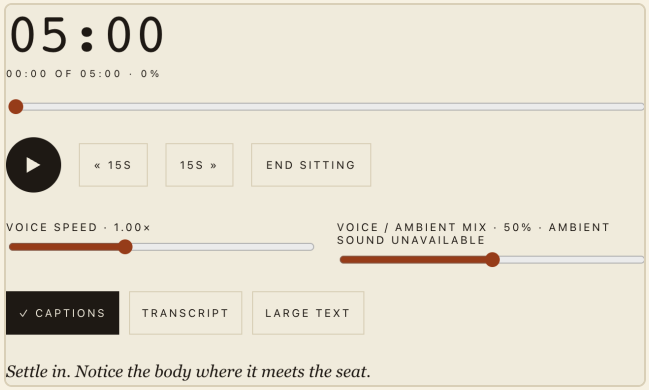
A hundred per cent, with a required step unfinished

Optional minutes fill the numerator and leave the denominator alone.

Executed evaluator case · pinned player crop

8 ÷ 7 → 100% displayed

Practice 4 and journal 4 done; required passage 3 skipped; treatment bonus.



Pinned player · a third percentage, at zero

Illustrative evaluator input; the meter and the completion flag answer different questions.

Displayed percentage · 100%

Completed estimated minutes over required minutes.

Required steps · 1 of 2

Count and ratio fail; the duration minimum passes.

ended_early

shouldCountAsCompleted false; shouldOfferRecovery true.

> Use shouldCountAsCompleted for the decision; the percentage answers a different question.

Three policies, three denominators

The optional-step treatment decides what the percentage divides by, and one of them cannot reach a hundred.

IGNORED

Required minutes only

Optional work enters neither side, so an all-optional ritual displays zero.

DENOMINATOR

required estimates, with a fallback when there are none

INCLUDED

Both sides grow

Optional estimated minutes join the numerator and the denominator.

DENOMINATOR

all required and optional estimates

BONUS · THE DEFAULT

Numerator only

Completed optional minutes count, and the denominator stays required.

DENOMINATOR

required estimates, with a fallback when there are none

One percentage, three policies, and a clamp that hides the difference at the top.

> Read the percentage policy and the completion policy together, or neither will make sense.

A skip has to be permitted by its step

Required and skippable are separate properties, and the helper throws when the step does not allow it.

A skip requested on the three-minute required passage of the running ritual.

Check Does the step exist and allow skipping?
Required and skippable are separate properties on the template step.

- yes → **Admitted** Return skipped progress
  Re-evaluate completion, choose the continuity reason, and stamp updatedAt.
- no → **Refused** Throw an operation error
  A missing step, or a step whose policy forbids skipping.

The helper returns data. The caller owns the event append, the save and the member feedback.

> A permitted skip returns changed progress and continuity; persistence still belongs to its caller.

Recovery selects what is required and unfinished

When nothing required is unfinished it falls back to anchors, which can reintroduce completed work.

Recovery for the running ritual, whose required passage was skipped.

Selection Is any required step unfinished?
Read the first matching progress entry for each required template step.

- yes → **Admitted** Retain those requirements
  Re-order them and mark every retained step required.
- no → **Try again** Fall back to the anchors
  Required, meditation or breathwork steps — completed ones included.

Both branches build a recovery template and a proposed plan; neither persists the original.

> A recovery result is a new proposal to inspect, not proof that the earlier record resumed.

What recovery proposes, and what it proves

A shorter template with the original policy, a stable completion object, and a link that routes rather than syncs.

Section 4 of 4

1 Three records, one word
A lifecycle contract, a domain evaluator and player telemetry, each deciding completed its own way.

2 What the session contract checks
An event with a consequence, a history that must agree with itself, and an audio floor no template lowers.

3 The evaluator, and the meter beside it
Required counts, a required ratio and an actual duration — and a percentage that answers none of them.

4 What recovery proposes, and what it proves
A shorter template with the original policy, a stable completion object, and a link that routes rather than syncs.

Each of these returns something well formed; none of them delivers anything.

A shorter list, the same seven minutes

The rebuild keeps the original estimate and minimum duration, so the retained step cannot satisfy them.

Illustrative case · executed helper behaviour

Seven minutes required for one three-minute step

Recovery retains the passage; the original eleven-minute estimate and seven-minute minimum survive.

Retained steps · 1

Only the unfinished required passage remains in the rebuilt template.

Duration minimum · 7 min

Carried into the rebuild unchanged, along with the eleven-minute estimate.

After three actual minutes

Count and ratio pass, duration fails, and the result is ended_early again.

An executed run of the current helper. No production fix and no successful recovery is asserted.

> Re-evaluate the returned recovery template; its friendly message is not a verified duration promise.

A completion event names the object

The identifier is the subject, the ritual and the occurrence, so a redelivery has to preserve the timestamp.

Source field specimen · no member data

tara.ritual.completed · source fields

Completed object	`objectId = userId + ritualId + occurredAt`	Stable for the same subject, ritual and occurrence — and only then.
Announcement identity	`eventId · eventType · source`	May differ between deliveries that describe the same object.
Completion evidence	`completionQuality · completedStepIds · skippedStepIds`	Quality, counts, step identifiers, minutes and reflection status.
Companion invitations	`handoffs · routePath · searchText`	Arete, Nisaba and Nyx, each with its own next action.

The displayed formula summarises the key fields; it is not a hash or a delivery receipt.

> A stable key supports replacement on replay only when the consumer preserves and enforces it.

Four projections, and no destination

The builder returns home, activity, search and recommendation objects, and stores none of them.

HOME AND ACTIVITY

Object-derived keys

Different action paths; activity takes the first handoff in the tuple.

CONSUMER RESPONSIBILITY

store, deliver and read back the intended view

SEARCH DOCUMENT

Subject-only audience

Filter version one, with a key derived from the completion object.

CONSUMER RESPONSIBILITY

enforce subject and current policy at query time

COMPANION CANDIDATES

Three, unsorted

Arete, Nisaba and Nyx with moment-based scores, in builder order.

CONSUMER RESPONSIBILITY

rank and admit them in the consuming surface

These functions return objects. No persistence, delivery, filtering or member result is asserted.

> A projection is useful data; delivery, authorisation and readback still need their own evidence.

A resume link routes; it does not sync

The token is built from the member, the plan, the week and the day, and it is not a credential.

A starter-plan continuation built on desktop, then reconstructed for mobile.

Compare Do the two supplied states match?
Device labels, the token, the plan, the session and the path.

yes →

Admitted Routing checks pass
The cards and links preserve the identity that was inspected, and nothing more.

no →

Refused Return validation issues
Name the mismatched routing evidence rather than a failed transfer.

No network or persistence call is made here; synchronised recovery is a separate claim.

> Matching links and a cross-device-ready flag do not establish synchronised progress.

Confirm the passage, and the saved return

The running case resolved, with the policy, the record and the destination each named separately.

Illustrative case · inspected library behaviour

“The passage is still your next step.”

Explain the remaining work and verify the returned policy before confirming continuity.

Keep the unfinished passage

The domain completion decision is still incomplete, whatever the meter says.

Inspect the recovery policy

The retained thresholds have to fit the return actually being offered.

Confirm the owning record

An authorised readback is what establishes that anything was saved.

A source-based review exercise, not a captured successful member recovery.

> A truthful return says what remains, what counts, and what has actually been saved.

What to carry out of this chapter

Three habits that outlast the specific thresholds: the record, the decision, and the proposal.

REMEMBER

1. **Name the record first**
   Contract, evaluator and player telemetry each own one meaning of completed.
2. **Read the flag, not the meter**
   shouldCountAsCompleted is the decision; the percentage answers another question.
3. **Inspect what a helper returns**
   Recovery spreads the original policy into a shorter template, and says it is shorter.

NEXT · CHAPTER 3 OF 5

Tara audio state and interruption

Next: Tara audio state and interruption — where the audio percentage this chapter's floor depends on comes from. →

> Every value in this chapter is exact, and every one of them decides less than its name suggests.