Tara service discovery, saved items and launch

The service-backed way into a practice, and the three boundaries it does not cross.

TARA · FIND A PRACTICE · CHAPTER 4 OF 5

Three things this chapter teaches, carried by one saved recommendation.

RUNNING EXAMPLE

One low-intensity recommendation — rec-low, with meditation med-low — found through canonical search, saved for later, and resolved into a launch descriptor that opens nothing.

LISTENING TIME

About 42 min · 31 slides

YOU WILL LEARN

1. **Three read paths, one name**
   A recommendation route, a canonical search, and the local engine from chapter three.
2. **Saving is a command**
   A desired state goes out; a reconciled view has to be asked for separately.
3. **A launch descriptor describes**
   It chooses an action, builds links and parses an intent, and it navigates nothing.

By the end you can name which path produced a result, what saving established, and what a launch descriptor is.

Three ways in, and who is admitted

Three read paths with different inputs, a guard before any of them, and the order pooling and filtering run in.

Section 1 of 4

1 **Three ways in, and who is admitted**
Three read paths, a route guard, role views that are one object, and the order of pooling and filtering.

2 **What a result actually says**
Field presence, three kinds of identity in one row, object kinds, freshness labels and a moment ordering.

3 **Saving, and reading it all back**
A command rather than a toggle, a reconciliation with a scope, and a bounded walk that needs a short page.

4 **Opening an item that nobody opened**
Action precedence, query overrides, a parser that describes, and the destination none of it visited.

> Trace the caller before explaining a result; three implementations answer to the same product word.

Find one, keep it, open it

The running example, and the three owners it passes through on the way.

Illustrative case · executed library and route behaviour

“Find a short practice and keep it for later”

Discovery, saved-item readback and the destination are three separate results.

Find a suitable record

Say which pools and which filters produced it.

Keep the meditation identifier

Send a desired state; reconcile a readback whose scope you know.

Resolve the destination

Inspect the action, the path, the query and the parsed intent.

Worked library and route behaviour; application consumers own the complete journey.

> A search result, a saved record and a launch descriptor each have a different owner and a different proof.

Three search paths, three inputs

They share a product word and read different things; only one of them forwards the member.

DIRECT RECOMMENDATIONS

User, limit, category, intensity

The route forwards all four to the service adapter.

OWNING SURFACE

the dedicated recommendation route

CANONICAL SEARCH

Pools, then filters

Recommendations always; courses and favourites need a configured member.

OWNING SURFACE

the dedicated search route

LOCAL WEB SEARCH

An inline array

Synonyms and weighted scores over twenty-eight declared records.

OWNING SURFACE

the search component of chapter three

Three separate source paths. Nothing connects them but the product word.

> Choose the actual caller before explaining a result, or the explanation will belong to another path.

Admitted before anything is read

The Tara guard runs after the abuse and authentication prehandlers and before any data operation.

A request has reached a dedicated Tara route.

The question the guard asks Does the context carry a Tara scope?
The Tara domain scope, or the wildcard.

yes →

Admitted Run the operation
A successful response is sent with a no-store cache header.

no →

Refused Refuse before reading
No context is unauthorized; a wrong scope is forbidden.

One request. The abuse and authentication prehandlers have already run.

Admission is decided first, and a refusal happens before a read rather than after one.

A role names a view, not a permission

Three roles declare different capability lists and receive the same runtime object.

	What the role declares Its capability list in the adapter.	What the role receives The object the registry hands back.
Shell	All eleven capabilities.	The shared adapter.
Admin	Seven; launch resolution is not among them.	The same object, launch resolver included.
Assistant	Nine, including saved items.	The same object; its TypeScript view omits saved items.
The role on a request	One of three names, or a bad request.	No extra permission check of any kind.

> Every role receives the same adapter instance, so the capability lists are documentation and the role name on a request is not a permission.

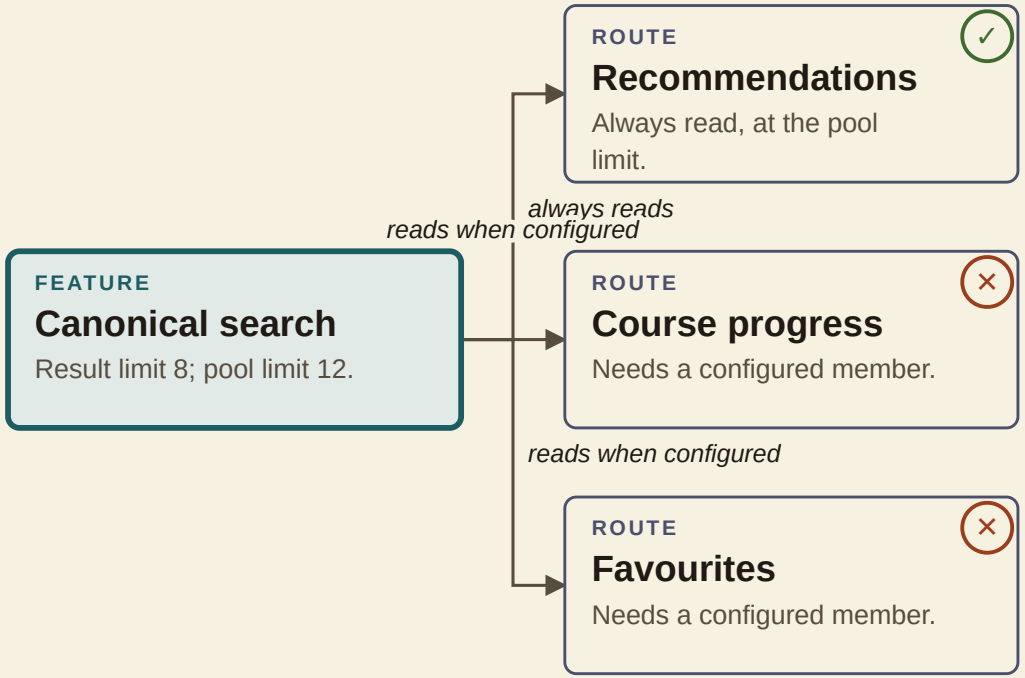
Three roles, answered for the declaration and for the object.

> A role selector describes an intended view; it does not remove a method or add an authorization check.

Which pools a search is allowed to read

Recommendations always; courses and favourites only when a member identity was configured.

Source configuration · conditional result pools



Knowledge Evidence

One configured adapter and the three reads it can start together.

> An include flag cannot supply a member identity that was never configured.

Build the pool, filter it, then cut it

Three steps in a fixed order, and the limit is applied to what survived rather than to what was read.

Canonical search · the three stages of one call

Build the combined records

Recommendation cards, then course cards, then favourite cards, mapped to results in that order.

Everything built crosses; no record is dropped at this edge. ↓ every mapped result

Apply every requested filter

Every requested filter has to pass: substring tokens, exact intensity, maximum duration.

Only survivors cross, and they keep their construction order. ↓ the survivors, in order

Take the requested count

The first n of whatever survived — the limit is applied to survivors, not to the pool.

One search. Each band hands the next everything that survived it, in the order it was built.

A small result limit does not move filtering ahead of construction, and nothing here ranks by relevance.

What a result actually says

Field presence decides eligibility, three identities share one row, and two labels describe rules rather than content.

Section 2 of 4

1 Three ways in, and who is admitted
Three read paths, a route guard, role views that are one object, and the order of pooling and filtering.

2 What a result actually says
Field presence, three kinds of identity in one row, object kinds, freshness labels and a moment ordering.

3 Saving, and reading it all back
A command rather than a toggle, a reconciliation with a scope, and a bounded walk that needs a short page.

4 Opening an item that nobody opened
Action precedence, query overrides, a parser that describes, and the destination none of it visited.

> Read each field by its own rule; one field in a result never validates another.

A filter reads a field, not an idea

An absent field is not a small value, and the same filter can exclude an entire result kind.

RECOMMENDATION

Duration and intensity

Both fields are populated on the card.

FILTER CONSEQUENCE

can pass either filter

FAVOURITE

Duration only

The intensity field is absent from the card.

FILTER CONSEQUENCE

an intensity filter excludes it

COURSE PROGRESS

Neither, but an estimate

A remaining-time estimate lives in another field.

FILTER CONSEQUENCE

either filter excludes the whole kind

Field absence is part of eligibility; a displayed estimate does not fill a missing search field.

> A course’s remaining-time estimate is not its duration, and a duration filter will not accept it as one.

Three identities in one row

What is displayed, what the canonical object names, and what the action targets are three strings.

Source field specimen · no member data

Worked recommendation · Calm breath

Displayed result	`tara-explore-recommended-rec-low`	Derived from the recommendation card.
Canonical content	`med-low`	Taken from the source recommendation’s meditation identifier.
Launch target	`/meditate/session/rec-low`	The action mapper uses the card’s session identifier.

Declared fixture record; resolving the destination’s content is owned elsewhere.

> Inspect the result record before treating any two of its identifiers as interchangeable.

What the object says, and what the action does

A ritual object still gets a session action, and a duplicate identifier still gets two rows.

	The canonical object What the builder constructs beside the row.	The mapped action What the action mapper returns for the card.
A ritual recommendation	Kind ritual, from the ritual or session identifier.	Still a session start, using the card’s session identifier.
A favourite	Meditation content, from the favourite’s meditation identifier.	A favourite play on a meditation-session path.
Course progress	Kind course, and no canonical object at all.	A course resume, preserving the current lesson.
Two rows, one identifier	The lookup finds the first matching record.	Both rows are emitted, each with its own title.

> The object and the action are built by different code from different fields, so a result can name one thing and target another.

Three declared record cases, answered for both columns.

> Object shape and action mapping are separate decisions, and neither repairs the other.

Freshness is a rule, not a measurement

Recommendations are marked new by their builder; the other kinds are labelled by a clock.

NEW

Six hours or less

Including a future timestamp, whose computed age is negative.

SCOPE OF THE LABEL

recommendations also receive it directly

RECENT

Six to forty-eight hours

From the course and favourite timestamp helper.

SCOPE OF THE LABEL

no claim about publication age

EVERGREEN

Older, or unreadable

Anything past forty-eight hours, and any invalid date.

SCOPE OF THE LABEL

an unparseable date lands here silently

A fixed clock in the fixture. These labels describe rules, not current member activity.

> A new badge says which branch of a helper ran, and not when any content was published.

A moment reorders the pool

A separate method scores intensity by the moment of day, and a zero only sinks an item.

Intensity score by ritual moment · configured · sort score; higher sorts first.

Moment	Low intensity	Medium intensity	High intensity
Morning opening	1	2	3
Sleep wind-down	3	2	0
Other moments	3	2	1

> A score of zero lowers a rank; it does not exclude anything, and it is not a safety decision.

Saving, and reading it all back

A command rather than a toggle, an identity the route supplies, and a reconciliation somebody has to apply.

Section 3 of 4

1 Three ways in, and who is admitted
Three read paths, a route guard, role views that are one object, and the order of pooling and filtering.

2 What a result actually says
Field presence, three kinds of identity in one row, object kinds, freshness labels and a moment ordering.

3 Saving, and reading it all back
A command rather than a toggle, a reconciliation with a scope, and a bounded walk that needs a short page.

4 Opening an item that nobody opened
Action precedence, query overrides, a parser that describes, and the destination none of it visited.

Sending a desired state is one result; knowing what the server holds is another, and it has a scope.

A desired state, not a toggle

True means add and false means remove; nothing is read first and nothing is inverted.

A caller has a meditation identifier and a state it wants to be true.

The question the helper asks Should this item be saved?
Read the supplied boolean; do not fetch and invert.

true →

Admitted Call add
The adapter’s response is returned, unreconciled.

false →

Refused Call remove
Reconciliation is a separate operation entirely.

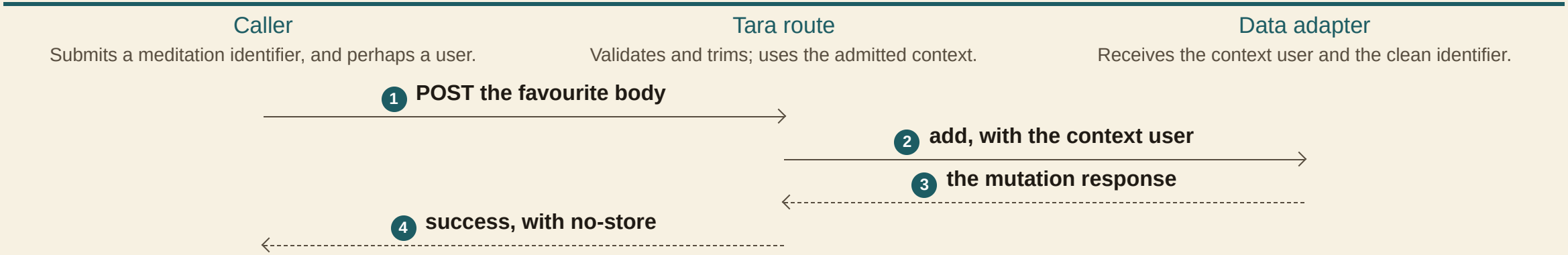
One call. Nothing was read before the question was asked.

> The helper does not read the current state, so the caller owns what the desired state should be.

The route supplies the identity

A body may claim a user; the route sends the authenticated one instead.

Declared authenticated favourite request · fixture response



One favourite mutation. The response is supplied by the fixture, not by a service.

> The authenticated context owns the user identity that reaches the adapter, whatever the request says.

Five fields, and four of them decide

The snapshot is a value the caller has to apply; timestamping it is not storing it.

Source field specimen · no member data

Tara favourite sync snapshot

The key	`meditationId`	One identifier is the whole lookup; it is not compared.
The compared fields	`title · duration · category · favourited`	Any of the four differing makes the row an update.
The returned clock	`lastSyncedAtMs`	Set on the returned value, which nothing has stored.

Added, updated, unchanged, removed and unreconciled counters come back beside the snapshot.

> Reconciliation returns a new snapshot and counters; adopting either of them is somebody else’s operation.

Absence means deletion only under one scope

A partial view preserves the local items it did not cover; a complete one removes them.

Reconciliation has processed the remote records and reached a local-only identifier.

The question the reconciler asks Was the remote view complete?
The acquisition path declares this; the reconciler cannot tell.

- complete →
  Admitted Treat it as removed
  Increment removed and leave it out of the new snapshot.
- partial →
  Try again Keep it, and say so
  Increment unreconciled and append the local record.

One local identifier, absent from the remote view. Additions and updates were handled first.

> A partial read cannot distinguish a deleted favourite from one it simply did not look at.

Two valid answers from the same two lists

Local A and B meet a remote updated A and a new C, and the scope decides what happens to B.

	A complete view The remote list covered everything.	A partial view The remote list covered some of it.
The returned order	Updated A, then C.	Updated A, then C, then local B.
A and C	One updated, one added.	One updated, one added.
Local B, uncovered	Counted as removed and dropped.	Counted as unreconciled and kept.
The local input	Unchanged; a new snapshot is returned.	Unchanged; a new snapshot is returned.

> Zero removals after a partial read means the uncovered items were kept, so the scope has to travel with the snapshot or the number is unreadable.

The same local and remote lists, answered for both scopes.

> Scope changes the treatment of the uncovered item, and nothing else about the reconciliation.

A failed page returns nothing at all

The walk exits before reconciling, and the snapshot the caller passed in is untouched.

The walk has read one or more pages and is deciding whether to reconcile.

The question the walk asks Did every attempted read resolve?
A rejection exits before the final reconciliation.

yes →

Admitted Return a reconciled snapshot
The caller decides how to apply and persist it.

no →

Refused Reject with no replacement
No partial snapshot, and no error message for anybody.

One page walk. The caller’s local snapshot is untouched on both branches.

Applying, storing and retrying a snapshot all belong to the caller, and so does saying that a read failed.

The walk that needs a short page

Two hundred at a time, twenty-five pages at most, and completeness is decided by a length.

Bounded offset walk · the helper’s own clock is its offset

offset 0	offset 200	any offset	offset 4,800
The first page	Each further page	A page shorter than two hundred	The twenty-fifth full page
Request two hundred records.	Two hundred more, with seen identifiers skipped on collection.	The remote list has been exhausted.	The cap is reached with every page full.
What this point decides	**What this point decides**	**What this point decides**	**What this point decides**
A raw batch, whose length is the only thing consulted.	A growing collected view, first copy of each identifier kept.	Scope complete; the loop stops here.	Scope partial; uncovered local items are preserved.

One walk along the offsets, with what each point decides.

> Reaching the page cap with full pages leaves the view partial, and the scope says so.

Where duplicates are handled, and where they are not

The page walk keeps the first copy of an identifier; the direct reconciler keeps both rows.

	Fetch and reconcile Collects pages, then reconciles.	Reconcile a supplied list Takes the remote rows as given.
A repeated identifier	Only the first occurrence is collected.	Both rows are processed in order.
A later copy with a different title	Discarded; the first title stays.	Kept; it can become a second snapshot entry.
The counters	One entry, counted once.	Unchanged can be incremented twice.
How the winner is chosen	First seen, without comparing timestamps.	No winner is chosen; nothing is compared.

> Duplicate handling is a property of the entry point rather than of the data, so the two paths disagree about the same rows.

One repeated identifier, followed through both entry points.

> Deduplication belongs to the collection step, so calling the reconciler directly bypasses it.

Opening an item that nobody opened

A bounded walk that needs a short page, first-seen duplicates, and a descriptor that describes.

Section 4 of 4

1 Three ways in, and who is admitted
Three read paths, a route guard, role views that are one object, and the order of pooling and filtering.

2 What a result actually says
Field presence, three kinds of identity in one row, object kinds, freshness labels and a moment ordering.

3 Saving, and reading it all back
A command rather than a toggle, a reconciliation with a scope, and a bounded walk that needs a short page.

4 Opening an item that nobody opened
Action precedence, query overrides, a parser that describes, and the destination none of it visited.

The last steps produce a well-formed description of a destination, and stop there on purpose.

Which card becomes which action

Four card cases, four action names, and two of them target a different identifier from the one on the card.

FAVOURITE

Play the favourite

The card’s session identifier was set from the meditation identifier.

PATH SHAPE

a meditation-session path

COURSE PROGRESS

Resume the course

A current lesson segment is preserved when the card has one.

PATH SHAPE

a course path, with a lesson when present

ORDINARY CONTINUATION

Resume the session

The session identifier is used directly.

PATH SHAPE

a meditation-resume path

RESTART, REFLECT OR REPLAY

Start from the meditation

The path uses the meditation identifier where one exists.

PATH SHAPE

a meditation-session path, with session metadata kept

Four card cases. Explore recommendations start a session; anything else falls back to an open.

> The action name and the identifier in its path are separate facts, and both belong to the card.

Action, then card, then path

Three inputs can describe a destination, and the first one present decides.

FIRST

An explicit action

If the caller supplied one, nothing after this is consulted.

WORKED RESULT

a supplied session start survives a competing card and path

THEN

A supplied card

Mapped to an action by the rules on the previous slide.

WORKED RESULT

the mapped action then owns the target path

LAST

A target path, or the hub

Only reached when no action was chosen at either tier.

WORKED RESULT

the Tara hub is the default destination

One resolution. The first tier that produces an action ends the choice.

> A competing target path does not override an action that is already there.

The query is spread last, and wins

Action metadata, origin and reference go in first; an explicit query replaces any of them.

Source field specimen · no member data

Executed override specimen

The returned action	`start-session`	Chosen from the explicit recommendation action.
The path-derived intent	`meditationId: rec-low`	Parsed from the target path, which did not change.
The explicit query	`action: custom · sessionId: other`	Spread last, so it replaces the metadata above it.

Origin and reference can be replaced the same way. The query is serialized as it stands.

> The returned query can disagree with the returned action and with the parsed intent, and all three are returned.

Four modes, read from the path alone

The parser normalizes a path and describes it; it never looks at the query metadata beside it.



A COURSE PATH

Decoded, with an optional lesson

Malformed percent encoding throws rather than falling back.

MODE AND OBJECT KIND

resume-course · course



A MEDITATE-RESUME PATH

The segment is the session

Used as the session identifier, as it appears.

MODE AND OBJECT KIND

resume · session



A SESSION OR RITUAL PATH

Not decoded at all

A ritual segment is copied into three identifier fields.

MODE AND OBJECT KIND

start · meditation or ritual



ANYTHING ELSE

Described as the hub

The normalized path is kept on the returned intent.

MODE AND OBJECT KIND

browse · hub

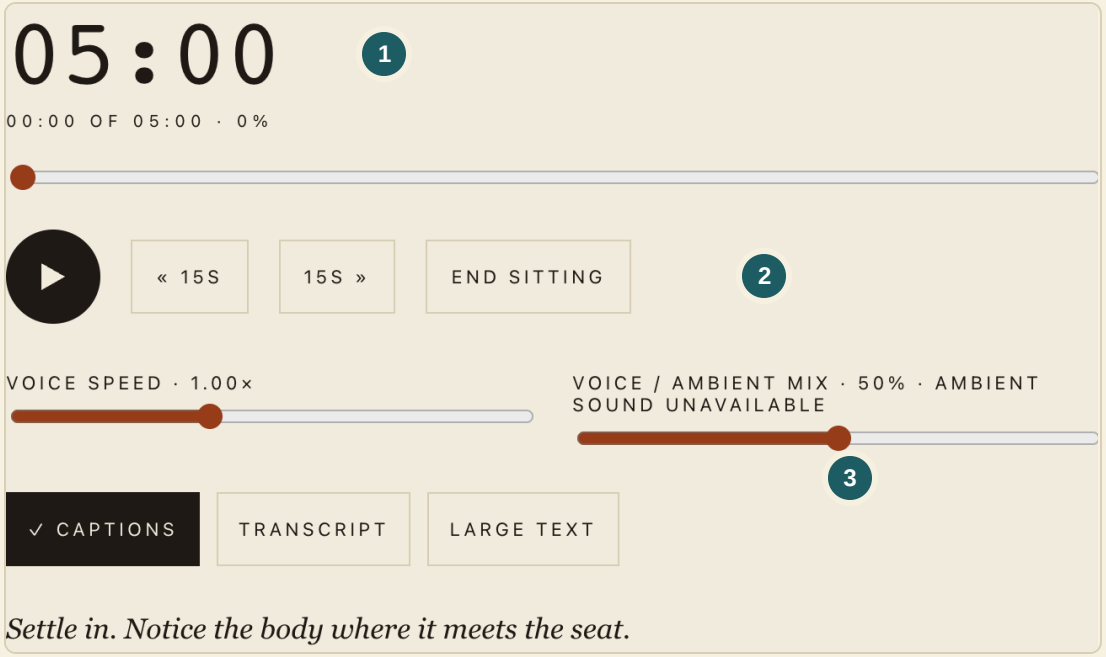
Four families after normalization. The query portion is removed before any of this.

> A parsed launch intent is a description of a path, and not an executed session.

What the path actually names

The descriptor describes a destination like this one, and has never been to it.

Implemented UI · source 1ceae52 · five-minute sitting, ambient refused



Tara sitting player · five-minute sitting · pinned component

Three things this destination shows, numbered in the order a reader should visit them.

1. **The destination does exist**
   A five-minute sitting, ready to start, at the kind of path the descriptor names.
2. **Nothing here pressed any of this**
   The resolver built an action, a path and a query; opening it belongs to the practise track.
3. **And it has failures of its own**
   The ambient sound request was refused, and the player says so rather than pretending.

> A resolved path is a string; the thing at the end of it is somebody else’s chapter and its own evidence.

Three owners, three proofs

Close on the running example and name the evidence each boundary would have to supply.

Illustrative case · executed library and route behaviour

“The result is saved — can I return to it?”

Keep the service record, the saved snapshot and the opened content connected by identifiers you checked.

Inspect the result’s identity

Which pool, which filters, which canonical object, which action.

Inspect the saved readback

Which user, which scope, and who applied the snapshot.

Inspect the destination

Which path was resolved, what it loaded, and what played.

Executed source mechanisms support these checks; live application acceptance is still separately required.

> A complete journey needs the service read, the saved-item readback and the actual destination — from three owners.

Tara service discovery, saved items and launch

Three things to keep, and where the track goes next.

REMEMBER

1. **Trace the caller first**
   Three implementations share a product word and read different inputs.
2. **Saving is a desired state**
   The readback is a separate operation, and its scope has to travel with it.
3. **A descriptor describes**
   An action, a target, a query and an intent — and nothing visited.

NEXT · CHAPTER 5 OF 5

Tara moods, themes and modalities

Moods, themes and modalities, and the teachers and lineages a practice is chosen from — chapter five, which closes the track. →

> Name the boundary before you name the outcome; this chapter is four of them in a row.