Integration Test Strategy

A template to adapt for your organization

Integration testing proves that components agree at their seams. It is the level most often assumed to be covered by somebody else, which is why the seam between two well-tested systems is such a reliable place for defects to live.

> ***How to use this template.*** *Everything in italics is guidance for you and should be deleted once you have replaced it. Square brackets mark a value to fill in. Work top to bottom the first time; after that, treat sections 3 and 4 as living documents and revisit them every quarter. Keep it short. A strategy nobody reads governs nothing.*

1. Scope

> *Name the seams, not the systems. "Order service to fulfilment" is a seam; "the order service" is not.*

#	Seam (A → B)	What crosses it	Synchronous?	Who owns each side
1	[Storefront → Order management]	[an order]	[no]	[team / team]
2				
3				

> Seams explicitly out of scope, and who covers them instead

2. What each integration test proves

> *Decide this per seam before building anything, because it determines the shape of the test and what a red result means.*

Claim	Worth building?	Notes
The hand-off works — the record created in A arrives in B, correctly and in time	Usually yes	The valuable one, and the one nobody owns
The whole journey works — end to end, every system	At most one	Goes red when any system has a bad day, and tells you little about which
Each system works — in isolation	Already covered	Necessary, and will not catch a hand-off failure

Build the first. Keep the third. Be deliberate before building the second.

3. Contracts and expectations

> *For each seam, write down what the receiving side is entitled to assume. Most integration defects are a violated assumption nobody wrote down.*

Seam	Field / behavior	Expectation	What happens if violated
[1]	[order total]	[matches the storefront to the cent]	[ ]
[1]	[currency]	[always present]	[ ]

4. Environments, stubs and the systems you do not control

System	Real, stubbed or sandbox?	Why	Who maintains it
[third-party payment provider]	[sandbox]	[cannot take real payments]	[ ]
[partner portal]	[stubbed]	[no test instance available]	[ ]

> Where you cannot reach the other side, **stop at the boundary**. Prove you handed off correctly, and prove you handle the response correctly, as two separate tests. Testing somebody else's uptime teaches your team to ignore red.

5. Timing across systems

Asynchronous hand-offs are where these tests actually fail. A test that checks system B immediately after system A will fail intermittently forever.

- Describe the **state** you are waiting for, never a fixed duration.
- Raise the wait on the **receiving** side, where the delay actually is.
- If a hand-off genuinely takes minutes, split it into two scheduled runs rather than holding a session open. Two honest results beat one flaky one.

> Expected hand-off latency per seam
>
> What we do when it exceeds that

6. Carrying a value between systems

> *The reference created in one system is what the next one searches for. Choose the mechanism deliberately — it is the single biggest factor in how robust these tests are.*

1. **Make the value predictable.** Have the test create a record it can identify without being told — a generated reference, a unique name. Then no hand-off is needed at all. Prefer this.
2. **Capture it and hand it on.** One step captures the value, a later step uses it.

3. **Fetch it in code.** When the value only exists somewhere a browser cannot see — a queue, an internal API, a mailbox.

> **With Functionize Studio:** each leg is a test inside the project for the system it happens in, and the journey is an orchestration that runs them in order. A red result names the leg, so it routes to the right team without anyone triaging it.

7. Automation candidates

> *Score each seam. Integration tests are more expensive to keep than single-system tests, so the business-risk column should carry more weight here.*

Flow	Business risk if it breaks (H/M/L)	Run frequency	Deterministic?	Data controlled?	Automate?
[seam 1 hand-off]					
[seam 2 hand-off]					

8. Entry and exit criteria

Gate	Criteria
Integration testing can start	[ ]
Integration testing is complete	[ ]
A seam is considered covered	[ ]

9. Ownership

A seam has two owners and therefore, by default, none. Name one.

Seam	Test owner	Reads a red result	Escalates to
[1]	[ ]	[ ]	[ ]
[2]			