INTEGRATIONS

Retrying a Failed Integration Should Not Duplicate the Order

You press send and an error appears. You press it again. Later, two identical orders appear in the destination tool. The first request arrived successfully; the response confirming it was the part that went missing.

Discuss this case when commissioning custom software. Demonstrations usually show data moving when everything works. Daily operations also involve interrupted connections, late responses and people trying again because the screen does not explain what happened.

Identify the Operation Before Sending It

Each operation needs a reference that distinguishes a retry of the same request. Use that protection if the destination supports it. Otherwise, agree another way to query and reconcile the result before repeating the request. The design depends on what both systems can do.

Make Uncertainty Visible

- Separate waiting to send, sent without confirmation and confirmed by the destination.
- Keep an attempt history with useful investigation references, without exposing credentials or unnecessary data.
- Limit retries and send cases that cannot be resolved automatically for review.
- Allow a case to be corrected without running every previous operation again.

During acceptance testing, simulate a late response, a connection lost after sending and several consecutive button presses. Check both the screen and the destination system. A friendly error message does not help if duplicate operations are left behind.

Specify who reviews pending work as well. A silent queue can accumulate tasks for days. Its owner needs to see what happened, which attempts were made and what decision is possible without creating another copy.

Duplicate prevention and recovery depend on the agreements between systems. Tools differ in how they confirm results, and some cases cannot be handled entirely through automation.

A Check You Can Run

In a test environment, let a sample order reach the destination and then interrupt the confirmation. Retry with the same operation reference and count the orders actually created. Stripe documents one implementation of idempotency keys, including how results are retained. Another provider may offer a different contract, so confirm its behaviour before adopting the same retry rule.

Supporting guidance: Stripe’s idempotent request contract.
https://docs.stripe.com/api/idempotent_requests

Continue with: Sold, Paid and Delivered Are Different Records.
https://systems.kavaliero.com/en/blog/sold-paid-and-delivered-are-different-records

For help with this work, see the scope of Custom Software.
https://systems.kavaliero.com/en/services/custom-software