John Fry | 609-865-0934
www.capmarketsme.com | Issued: 06/27/2026

Workflow Upgrade Testing

Components First, Not End-to-End

A workflow looks like one big complicated thing, so people try to test it as one big complicated thing. Put a little code on the transitions, and the number of routes through even a ten-state workflow runs into the thousands; you can realistically write forty or fifty test cases. So end-to-end testing checks about one percent of the routes, and you choose that one percent by guessing at risk, usually without quite realizing that is what you are doing.

- **What this is.** A way to test a workflow upgrade by testing the parts that can actually change — the transition code and the gates, meaning the rules and filters — instead of replaying trades end to end.
- **Why you should care.** End-to-end covers about one percent of the routes, picked by guesswork. Component testing covers only the parts an upgrade can touch, which tends towards the whole of what matters. The engine does not change, and the configuration is carried across unchanged, so there is nowhere else for behavior to hide.
- **Your options.** Test end-to-end and live with one percent. Or test each component, show it behaves the same before and after the upgrade, and allow the risk on the rest to get to zero — where it really is. Two real choices, and one of them is mostly hope.
- **What happens if you choose wrong?** You ship an upgrade you believe is tested. You have checked one percent, chosen by a risk estimate you cannot defend, under a fixed deadline, where quality is the only thing that is flexible, so quality shrinks. The failure then turns up in production, which is the one place you did not want it.
- **The catch.** None of this works if you do not understand the architecture well enough to believe it. You do not retest that RAND() still works when Excel upgrades; you trust the product and test your own formulae. This is the same. Believing it takes a real grasp of how the workflow is built.

Two things follow, and I will be plain about both. To believe any of this you have to understand the architecture — the engine, the configuration, the gates, and how they work together. That is a separate paper, and you need it because the argument gets stronger once you can see why the structure is a fixed point. Actually doing the testing — the inventory, the read/write map, the differential set, the rule and filter scripts — is separate work again, and that is the engagement, not the argument. This page makes one claim and only one: you cannot test a workflow end-to-end.