Fabric vs. Custom Code: Simplifying Hybrid Integration

How IT teams are ***saying yes faster*** with Adaptive Integration Fabric.

CHALLENGE	CUSTOM SCRIPTS & CODE	SAY YES FASTER *with Adaptive Integration Fabric*
Development Time	Long timelines due to custom coding, brittle logic, and coordination across teams	→ Drag-and-drop design studio reduces build time by up to 80%; deploys to production in days
Reuse & Scalability	Logic recreated project by project; prone to inconsistencies and high testing overhead	→ Reusable, version-controlled components standardize behavior and shorten test cycles
Skill Requirements	Requires deep COBOL or PL/1 knowledge; limited to backend specialists	→ Built for generalist developers; no-code environment eliminates need for low-level access
Integration Types	Difficult to orchestrate inbound/ outbound workflows across systems	→ Supports REST, SOAP, mainframe subroutines, external APIs, and hybrid flows from one interface
Error Handling	Manual patching and post-deployment fixes introduce risk and delay	→ Flexible input enforcement, runtime validation, and detailed logs enable proactive debugging
API Standards	No automatic alignment with modern API specs	→ Native OpenAPI support including operation IDs, input type mapping, and auto-exported specs
Change Management	Fragile updates and complex change requests slow response to business demands	→ Visual change tracking, comment annotations, and wizard-based project creation simplify updates
Security & Governance	Scripts often lack centralized control or auditability	→ Centralized orchestration with configurable user roles and trace logging for oversight