



# Integration-Ready Ecommerce: How to Build for Seamless Connections

Everything you need to prevent fragile,  
failure-prone integrations in mature  
platforms

Plugging in new systems isn't the hard part. Keeping your platform from cracking after they're live — that's where things get real. In mature ecommerce systems, integrations collide with years of architectural decisions: legacy modules, hidden dependencies, embedded business logic, and workarounds built under pressure.

That's why this whitepaper isn't about how to wire one system to another. It's about how to prepare enterprise platforms for integrations that won't collapse under scale or complexity.

You'll get a hands-on breakdown of what makes integrations fail in mature systems and what to do about it. The whitepaper walks through concrete strategies to prepare your architecture and avoid costly missteps before the next integration begins.

Every insight comes from what's worked inside high-load ecommerce systems built and scaled by Expert Soft. These are patterns shaped in production, not in theory — ready for teams who can't afford to get integrations wrong.

# Table of Contents

|                                                        |    |
|--------------------------------------------------------|----|
| Why "Being Ready" Beats "Being Fast"                   | 3  |
| Symptoms That a System Needs Tuning                    | 4  |
| Making Your System Integration-Ready                   | 7  |
| AI Readiness as the Next Level of Integration Maturity | 11 |
| Integration Readiness Checklist                        | 13 |

# Why “Being Ready” Beats “Being Fast” in High-Stakes Integrations

In mid-size systems, integration might mean wiring up a few services and tweaking endpoints. In high-load enterprise platforms, it's a different story. These systems carry years of architectural weight: legacy modules, business logic embedded deep in controllers, and workaround-layered solutions built under pressure.

**//** *All these factors may turn even a straightforward API connection into a potential risk point. //*

Yet business expectations don't scale down with system complexity, maintaining the pressure to move fast.

**//** *It often drives teams to treat integrations like sprints — just plug in, test, deploy — ignoring how enterprise platforms actually behave. //*

But the only result this path can lead to is endless firefighting and technical debt that only grows.

That's why enterprises should “be ready” for integrations, meaning resisting shortcuts: refactoring what's brittle, decoupling what's tightly bound, and setting the ground for integrations that won't collapse under scale.

That's exactly what a global medical equipment manufacturer — an Expert Soft's client — did when replacing their SAP Commerce back-office with a custom PIM system. Built with Sitecore and micro frontends, it connected to SAP Commerce via REST APIs and enabled product deactivation, attribute copying with language context, and API-driven reporting — all mapped to real business needs.

Instead of piling on complexity, they restructured from the core, gaining a clean UI for business users and stable, integration-ready services underneath.

In high-stakes environments, readiness means flexibility, visibility, and structural clarity — not just delivery speed.

Want more than technical fixes?  
Find stories, systems, and strategies  
that scale on LinkedIn.

JOIN!



# Symptoms That Indicate Your System Needs Tuning to Become Integration-Ready

If core architecture is unstable, even the best-planned integration can trigger unexpected failures. These are real-world symptoms we've seen in our practice that signal a system is not yet ready for scalable and safe integrations.

## SYMPTOM 1: BUSINESS LOGIC HARDWIRED INTO INTEGRATION CODE

When integration layers carry business rules, such as validations, fallbacks, or conditional flows, they stop acting as connectors and start behaving like brittle extensions of the core system.

This breaks the separation of concerns and makes each change a risk. Instead of scalable, plug-and-play architecture, teams end up debugging business logic embedded in glue code.

### WHAT CAN GO WRONG

- In a leading Baltic beauty retailer's system, gift promotions were misapplied because cart evaluation logic had to be custom-encoded in client systems.
- A [global luxury ecommerce platform](#) had to embed client-side workarounds in its payment flow to handle unstable logic from a regional payment provider — iyzico, making the integration fragile and hard to maintain.
- A European cosmetics chain ran into critical issues integrating a third-party review engine when its scripts clashed with Angular components, forcing in-code overrides that jeopardized front-end stability.

## SYMPTOM 2: UNCLEAR SYSTEM-TO- SYSTEM DEPENDENCIES

Enterprise integrations depend on clean, well-mapped relationships between systems and processes. When those dependencies are undocumented or overly coupled, changes in one component can unintentionally break others.

**//** *Without clear visibility into data flows and integration points, teams are left guessing – and guessing at scale is expensive.* **//**

### WHAT CAN GO WRONG

- I** In a major European health and beauty retailer, changes in a delivery provider's API broke the checkout flow when response formats shifted unexpectedly, revealing a lack of defensive handling in upstream dependencies.
- I** In a major luxury ecommerce rollout, missing documentation in the payment integration forced teams to implement duplicate logic paths, resulting in unpredictable order states and post-checkout failures.

## SYMPTOM 3: INCONSISTENT OR FRAGMENTED DATA

Successful integrations depend on unified data, but when systems operate on fragmented versions of product or customer information, integration logic becomes unreliable. APIs return mismatched content, personalization engines make poor decisions, and synchronized processes desynchronize at scale.

Without centralized governance, each new connection only amplifies the inconsistency.

### WHAT CAN GO WRONG

- I** In a major European cosmetic retailer, outdated pricing and imagery across multiple caching layers caused API-fed storefronts and third-party tools to operate on stale data, breaking downstream consistency across integrated channels.
- I** In a multinational cosmetics and perfume brand, a promotions engine failed to synchronize updates across integrated services due to asynchronous cache refresh, leading to inconsistent campaign behavior.

## SYMPTOM 4: SYNCHRONOUS-ONLY THINKING

Many ecommerce systems rely on synchronous interactions, assuming immediate availability and response from third-party systems.

// *But resilient integrations need asynchronous, event-driven architectures, where services communicate through queues or events, not direct calls. //*

This decouples dependencies and allows systems to absorb spikes, delays, or failures without breaking the user flow.

### WHAT CAN GO WRONG

- I A [global financial intelligence provider](#) relied on a nightly sync between MySQL and MongoDB to feed data into integrated search and analytics platforms. When syncs failed or delayed, downstream systems received outdated data, breaking real-time reporting and introducing inconsistencies.

## SYMPTOM 5: MISSING OBSERVABILITY INTO WHAT FAILS AND WHY

Even the best integrations fail at times, but without observability, teams are left blind. When logs are missing, tracing is incomplete, and alerts arrive too late (or not at all), diagnosing integration issues becomes guesswork. Failures silently propagate across services, consuming hours of investigation and increasing the risk of unresolved errors in production.

### WHAT CAN GO WRONG

- I A European beauty brand lacked proper audit trails, making it impossible to trace unauthorized changes in production environments. This turned minor incidents into major integration outages.
- I Before introducing targeted logging, a leading retail group struggled with silent data loss during order processing that was impossible to catch.

## ABOVE THE LIST

Monitor the system's caching, as when multiple systems cache the same data independently, or when invalidation flows aren't aligned, integrations become unreliable. Each layer might serve outdated content, conflicting responses, or inconsistent states across connected services.

Explore more ways how bad caching can [undermine your ecommerce system](#)

# Making Your System Integration- Ready

Integrations shouldn't feel like defusing a bomb. If your architecture is solid, each new connection becomes routine — not a rescue mission. Here's what to build into your system now to make future integrations fast, stable, and drama-free.

## 1. BREAK THE MONOLITH BEFORE IT BREAKS YOUR INTEGRATIONS

### WHEN YOU SKIP:

Legacy monoliths often contain entangled logic and shared states, making integration risky.

### ACTION:

Reduce the blast radius by decoupling legacy modules and extracting bounded contexts into modular services.

### IMPLEMENTATION TIPS:

- Introduce anti-corruption layers between legacy modules and modern services to isolate outdated assumptions.
- Extract bounded contexts into dedicated services (e.g., inventory, pricing, promotions).
- Use feature toggles and strangler patterns to incrementally route traffic to new modules.

## 2. ALIGN YOUR DATA BEFORE YOU INTEGRATE ANYTHING

### WHEN YOU SKIP:

Fragmented product and customer data leads to mismatched pricing, broken personalization, unreliable search results, and conflicting analytics across integrated systems.

### ACTION:

Establish strong data ownership and enforce consistency through synchronized, validated sources of truth.

#### IMPLEMENTATION TIPS:

- Define a single source of truth for key datasets like product catalogs and customer profiles.
- Sync data in real time using Kafka or CDC tools (e.g. Debezium).
- Apply data validation and enrichment rules before exposing data to external systems.

### REAL-WORLD INSIGHT:

In a seller of luxury accessories, centralized product tagging ensured consistent behavior across AI recommendation engines and faceted search, reducing integration overhead.

## 3. LEVERAGE ASYNCHRONOUS, EVENT-DRIVEN ARCHITECTURE

### WHEN YOU SKIP:

Synchronous-only communication creates tight coupling between services, slows down response times, and amplifies the risk of cascading failures when integrated systems stall or lag.

### ACTION:

Adopt event-driven patterns to decouple service interactions, improve fault tolerance, and scale integrations with less friction.

#### IMPLEMENTATION TIPS:

- Use event brokers like Kafka or RabbitMQ to separate producers from consumers.
- Apply idempotency keys and retry strategies to handle transient failures.
- Design services to emit clear domain events, such as OrderPlaced or ProductBackInStock.

### REAL-WORLD INSIGHT:

A major health and beauty retailer implemented event-driven pipelines to feed real-time data into an AI-powered keyword engine for dynamic navigation. This decoupled the search experience from back-end systems and made integration more flexible and scalable.

## 4. MAKE CACHING WORK WITH YOUR INTEGRATIONS

### WHEN YOU SKIP:

Poor cache governance leads to stale or conflicting data across services. Integrations that rely on fresh product info, personalized content, or updated configurations begin to break in unpredictable ways.

### ACTION:

Build a caching strategy with clear ownership, invalidation rules, and traceable freshness to avoid hidden integration issues.

#### IMPLEMENTATION TIPS:

- Define which service owns and invalidates each cache.
- Use centralized cache invalidation strategies (e.g., event-driven purge).
- Implement cache versioning or ETag-based validation for precision control at the edge.

### REAL-WORLD INSIGHT:

A large European retail brand had to re-engineer its caching logic to support CMS-driven campaign personalization. This ensured accurate targeting and consistent content delivery across integrated services.

## 5. BUILD IN OBSERVABILITY TO CATCH FAILURES EARLY

### WHEN YOU SKIP:

Integration issues often go undetected until they hit the customer. Without visibility into how systems communicate, failures become harder to trace, slower to fix, and more costly to resolve.

### ACTION:

Build observability into every integration layer to surface failures early, understand impact fast, and recover without firefighting.

#### IMPLEMENTATION TIPS:

- Use distributed tracing (e.g., OpenTelemetry, AWS X-Ray) to track requests across services.
- Correlate logs, metrics, and alerts using tools like Datadog, Prometheus, or ELK.
- Monitor business-level signals, such as failed checkouts or unprocessed payments, not just infrastructure metrics.

### REAL-WORLD INSIGHT:

A global financial data provider we mentioned in the previous section rolled out full-stack observability to monitor critical data migrations between MySQL and MongoDB. This ensured continuity across integrations with zero customer disruption.

## 6. STANDARDIZE INTEGRATIONS WITH REUSABLE FRAMEWORKS

### WHEN YOU SKIP:

One-off integrations multiply tech debt and make maintenance unpredictable. Each new connection requires re-implementing the same logic – authentication, retries, logging – wasting time and increasing risk.

### ACTION:

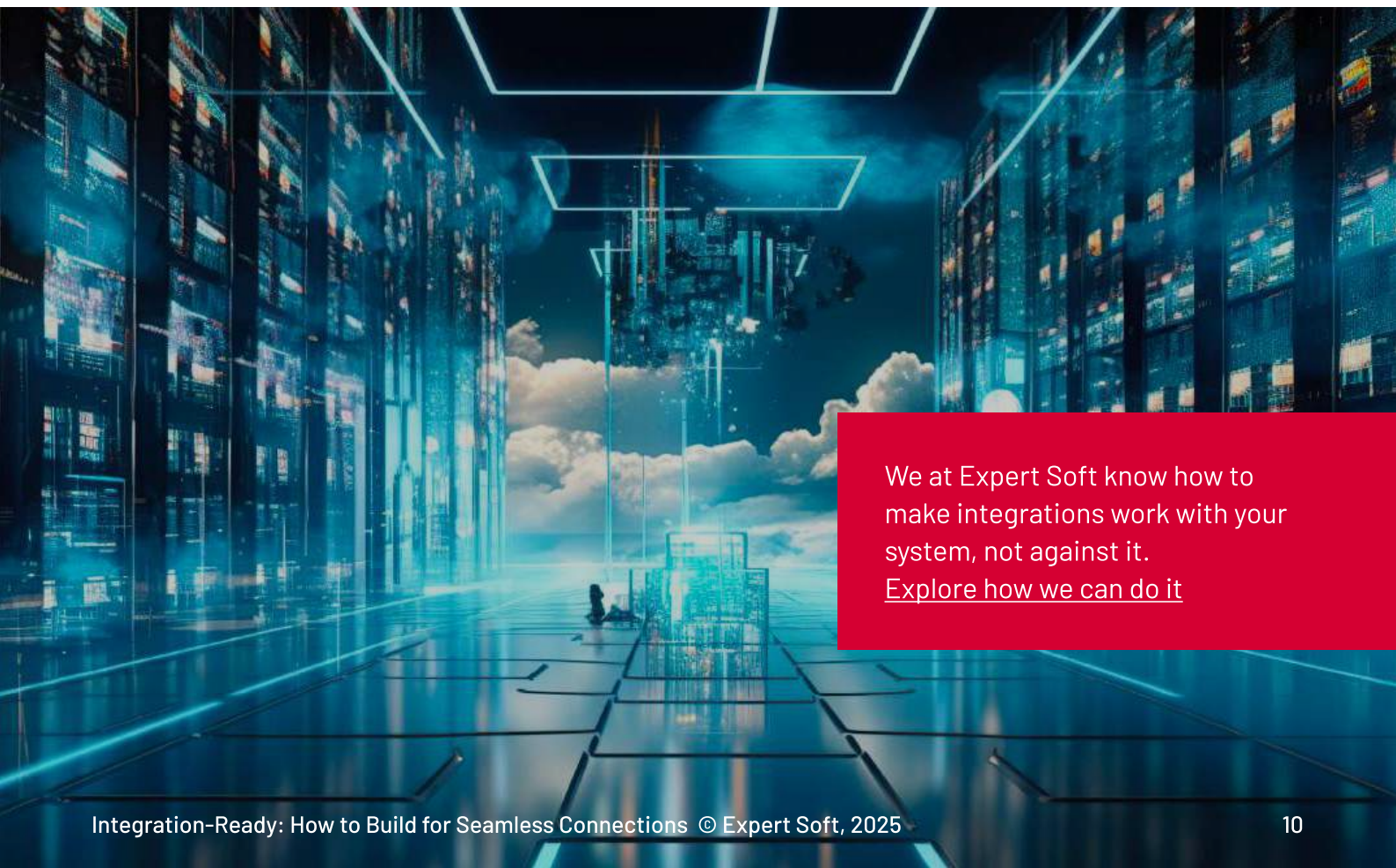
Build reusable libraries and shared contracts to make integrations faster, testable, and consistent across the platform.

### IMPLEMENTATION TIPS:

- Publish interface contracts using OpenAPI specs or GraphQL introspection.
- Use mocked environments to automate regression tests for third-party systems.

### REAL-WORLD INSIGHT:

A world-know ecommerce platform built a reusable integration framework for payments, bundling retry logic, fraud flagging, and logging. This significantly cut onboarding time for new payment providers and reduced error rates in production.



We at Expert Soft know how to make integrations work with your system, not against it. [Explore how we can do it](#)

# AI Readiness as the Next Level of Integration Maturity

As AI capabilities continue to spread across industries, including ecommerce stacks, we can't ignore the topic. In fact, it deserves a closer look, because AI integrations are more capricious than traditional ones: less forgiving, more data-hungry, and extremely sensitive to input quality.

To deliver real value, AI needs structured, real-time, high-volume data that's clean, enriched, and consistent. Anything less, and the output turns into noise or outright failure.

There are several factors that make AI integrations especially demanding:

## **REAL-TIME DATA ACCESS**

AI systems, such as recommendation engines or fraud detection models, must operate on the most current transactional and behavioral data.

## **STABLE AND SCALABLE APIS**

AI services, especially those deployed as microservices or external endpoints, must operate under unpredictable loads. If APIs slow down or break, so does the intelligence layer.

## **CONSISTENCY OF INPUTS AND OUTPUTS**

AI models depend on tightly defined inputs. Miss a field, rename an attribute, or shift a format, and the system either errors out or delivers poor predictions. And it's not just about inputs. Always keep an eye on the outputs too, because even the most stable models can produce flawed results without warning.

## **UX-LEVEL INTEGRATION**

AI solutions often require deep coupling with front-end logic — dynamic content, smart search, virtual try-ons — meaning integration requires tight backend-frontend coordination.

## **WORKFLOW COMPATIBILITY**

AI tools for translation, tagging, or moderation must live inside real business processes. Dropping them into production without aligning with editorial and publishing flows leads to bottlenecks or total rejection.

When these conditions are unmet, even advanced AI services deliver subpar results. Worse, they may increase complexity without generating proportional value.

## WHAT IT TOOK TO DELIVER AI IN PRACTICE

The things mentioned are exactly what we at Expert Soft keep in mind when integrating AI solutions for our clients. Here are some examples of how we've had to tweak systems to ensure smooth and reliable AI capabilities.

### **AI-DRIVEN VIRTUAL TRY-ON**

To enable users to try cosmetic products on the website, we had to set up clean interactions between front-end logic and back-end services, ensuring low-latency data access and consistent state management.

### **DYNAMIC AI KEYWORD ENGINE FOR AUTOMATED CATEGORY GENERATION.**

This required tight integration with product tagging and filtering rules to ensure relevancy, consistency, and immediate reflection of catalog updates.

### **AI-BASED CONTENT TRANSLATION TO REPLACE MANUAL LOCALIZATION.**

Replacing manual processes with AI-powered translation reduced operational costs but required CMS reconfiguration and workflow adjustments to integrate AI into content creation and publishing cycles without breaking quality standards.

### **AI-POWERED SEMANTIC SEARCH WITH MULTILINGUAL AND GEO-BASED ACCESS CONTROL**

To support natural language queries and enforce regional content restrictions, we built a RAG-based vector search system. The ecommerce platform was adjusted to expose consistent, structured metadata across services and enforce geolocation and user-role filters at the integration level, ensuring seamless, compliant communication with the AI engine.

Get inspired by [real-world AI in action](#) and see how ecommerce enterprises are turning bold AI ideas into powerful features

# Integration Readiness Checklist

By now, one thing should be clear: successful integrations are more than just wiring systems together. They're about building the kind of internal structure that doesn't collapse under pressure. The more complex your ecosystem, the more unforgiving the cracks become.

Before your next integration, run your system through this list to assess whether your system is truly integration-ready.

- ☐ **Is your business logic decoupled from integration glue code?**  
If critical decisions live inside API bridges or custom adapters, flexibility is already compromised.
- ☐ **Are system-to-system dependencies clearly documented and loosely coupled?**  
One change shouldn't trigger a chain reaction across unrelated modules.
- ☐ **Do all services rely on consistent, centralized product and customer data?**  
Fragmented inputs mean broken personalization, bad pricing, and unreliable analytics.
- ☐ **Are cache layers coordinated and predictable across all connected systems?**  
Misaligned cache invalidation leads to stale content, broken promotions, or inconsistent UX.
- ☐ **Is your platform structured to support reusable, testable integration components?**  
If every integration is a one-off, you're burning time and adding fragility.
- ☐ **Are AI capabilities built on data that's fast, clean, and structured?**  
AI doesn't forgive messy inputs. If your system isn't feeding it right, it's not helping you.
- ☐ **Can you trace, log, and alert on integration failures in real time?**  
Observability isn't optional. If something breaks and you don't know instantly, it's already too late.

If your checklist has a few empty boxes, don't worry — we at Expert Soft build ecommerce platforms that turn legacy blockers into integration-ready machines. [Explore how](#)

# About Expert Soft

Expert Soft is a targeted ecommerce software delivery company, partnering with Fortune 500 companies and global corporations across the US and EU. With SAP Commerce Cloud and Java as our backbone, we know how to ensure scalable and high-performing solutions that can handle 1 mln requests per second, delivering a smooth customer experience.

Developing a payment engine that saved our client about \$100 million in operational expenses, ensuring multi-country platform support, adapting solutions for new market entry with tailored enhancements — these are just a few of the challenges our specialists tackle.

We aim to deliver more than a software system. We aim to deliver tailored solutions that maximize profitability within available resources. Our success is driven by:



## TEAM STRENGTHS

- | All our engineers have a university background
- | Specialists excel their skills in our training LABs
- | Perfect English skills
- | Ready to help 24/7

### CLIENTS

We work with corporations around the world with revenue of over \$20 billion and 150K+ employees.

### APPROVALS BY AUDITS

Our ongoing work with corporations is consistently validated through rigorous audits, both by internal teams and Big 4 consulting firms.

### HIGH-LEVEL SECURITY

Approved by assessments from global companies, who are leaders in their respective industries.

### BUDGET EFFICIENCY

By carefully aligning technology investments with your business goals, we ensure optimal value and cost-effectiveness.

### PROFESSIONAL TEAM

No offshore outsourcing and our team's average tenure of 4+ years means you get seasoned problem-solvers, not just coders.

## EXPERT SOFT EXCELS IN

- | PAYMENT ENGINE
- | MICROSERVICES ARCHITECTURE
- | CONTENT MANAGEMENT
- | REDESIGN
- | E-COMMERCE PLATFORM
- | HEADLESS COMMERCE
- | MICRO UI FRONT-END
- | MIGRATION&INTEGRATION

## OUR TECH CORE



### FRONT-END

HTML, CSS, JavaScript  
(Angular, React, Vue,  
Next, TypeScript,  
Jquery), Spartacus



### BACK-END

Java EE, Spring, SAP  
Commerce (Cloud),  
Node.JS.



### DEVOPS

Docker, Kubernetes, CI/  
CD



### UX/UI DESIGN

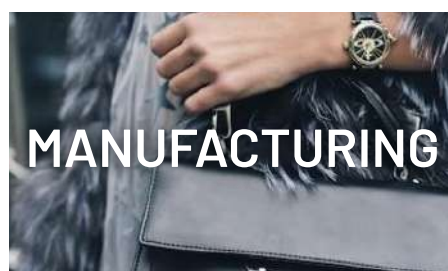
UX Research, UI Design,  
Figma, Adobe, Sketch



### QUALITY ASSURANCE

Manual Testing, Test  
Automation

## TARGETED DOMAINS



## SHARED PATHS, LASTING ECOM VICTORIES



## LET'S TALK SOLUTIONS!



**EKATERINA LAPCHANKA**

Chief Operating Officer

[kate.lapsenco@expert-soft.com](mailto:kate.lapsenco@expert-soft.com)

[+1 585 499 7879](tel:+15854997879) 

[+371 25 893 015](tel:+37125893015) 



**PAVEL TSARYKAU**

CEO & Founder  
of Expert Soft

Let's connect 