Here are some high-level solutions to keep service interfaces (APIs/events) in sync:

Design-first, single source of truth

Define contracts in an IDL (OpenAPI/AsyncAPI/Proto/Avro) and treat them as code (in repo, PR-reviewed); auto-generate server stubs, client SDKs, and docs from that spec so code can’t drift.

Compatibility-first versioning policy

Prefer additive, backward-compatible changes; use semantic versioning; allow v1/v2 to coexist behind the gateway; publish deprecation windows (Sunset headers/changelogs) and remove only after telemetry shows near-zero usage.

Consumer-driven contract testing in CI

Use CDC (e.g., Pact/Spring Cloud Contract) so each consumer publishes expectations that providers must pass; add schema-evolution checks to block breaking changes before merge.

Gradual rollout with adapters

Route by version/header at the API gateway; use shims/transformers to translate old↔new payloads; canary and shadow traffic; dual-write/read patterns to keep old and new models aligned during migrations.

Runtime governance & visibility

Emit metrics on contract violations (unknown fields, validation errors); log business IDs and versions; track endpoint/field usage to guide deprecation; keep living docs and a clear comms cadence for interface changes.