Architecture Decision Records

ADR-014 — Hybrid Encryption for L4 Restricted Fields

ADR-014 — Hybrid Encryption for L4 Restricted Fields (PIB, JMBG, OIB, JIB, IBAN)

> **ADR Number:** ADR-014 **Title:** Use AES-256-GCM field-level encryption for JMBG and OIB; rely on disk-level encryption + application-layer access controls for PIB, JIB, and IBAN **Date:** 2026-02-25 **Author:** Petter Graff (Architect) **Status:** Proposed **Resolves:** Conflict between SECURITY-ARCHITECTURE.md (2026-02-20, "Column-level encryption: Not needed") and DATA-ENCRYPTION-POLICY.md / COMPLIANCE-FRAMEWORK.md (2026-02-23, "AES-256-GCM field-level encryption MANDATORY for all L4 fields")

---

1. Context

1.1 Situation

Bilko is a cloud accounting SaaS targeting 50K-500K SMBs across Serbia, Bosnia & Herzegovina, and Croatia. The database stores several categories of personal and financial identifiers classified as L4 Restricted:

Field	Identifier Type	Jurisdiction	Nature
JMBG (Jedinstveni maticni broj gradana)	Unique citizen number	RS, BA	**Personal** -- encodes date of birth, gender, region of birth. Equivalent to SSN. Assigned to natural persons only.

Field	Identifier Type	Jurisdiction	Nature
OIB (Osobni identifikacijski broj)	Personal identification number	HR	**Personal** -- assigned to every natural and legal person in Croatia. 11-digit number used across all government and financial interactions.
PIB (Poreski identifikacioni broj)	Tax identification number	RS	**Business** -- assigned to legal entities and entrepreneurs for tax purposes. Publicly visible on invoices, SEF portal, APR registry.
JIB (Jedinstveni identifikacioni broj)	Unique identification number	BA	**Business** -- assigned to legal entities for tax purposes. Publicly visible on UIO portal, court registry.
IBAN	International Bank Account Number	All	**Financial** -- bank account identifier. Semi-public (printed on invoices, shared for payment).

1.2 The Conflict

Two Bilko security documents contradict each other:

SECURITY-ARCHITECTURE.md (2026-02-20):

> "Column-level encryption: Not needed (disk encryption sufficient for accounting data)"

DATA-ENCRYPTION-POLICY.md and COMPLIANCE-FRAMEWORK.md (2026-02-23):

> "AES-256-GCM field-level encryption MANDATORY for all L4 fields (PIB, JMBG, OIB, JIB, IBAN)"

The earlier document (Security Architecture) was written from a pragmatic engineering perspective. The later documents (Encryption Policy, Compliance Framework) were written from a compliance/DPIA perspective and prescribed a blanket field-level encryption mandate for all L4 fields without differentiating between personal identifiers and business/financial identifiers.

1.3 Regulatory Analysis

GDPR Article 32 -- Security of Processing (Croatia)

GDPR Article 32 requires "appropriate technical and organisational measures to ensure a level of security appropriate to the risk," taking into account:

1. The state of the art
2. The cost of implementation
3. The nature, scope, context, and purposes of processing
4. The risk of varying likelihood and severity for the rights and freedoms of natural persons

Encryption is listed as one example ("inter alia as appropriate: the pseudonymisation and encryption of personal data") but is **not a blanket requirement**. The determination is risk-based.

GDPR Article 87 -- National Identification Numbers

Article 87 permits Member States to establish specific conditions for processing national identification numbers. It requires "appropriate safeguards" but does **not mandate encryption specifically**. Croatia (through AZOP) has not published binding guidance requiring field-level encryption of OIB in databases, though OIB processing is subject to purpose limitation and data minimization principles.

Serbia -- ZZPL (Sl. glasnik RS 87/2018)

Serbia's ZZPL is closely aligned with GDPR. Article 50 (equivalent to GDPR Art. 32) requires "appropriate technical, organizational and personnel measures." The Serbian Commissioner (Poverenik) has not published specific guidance mandating field-level encryption for JMBG or PIB. However, JMBG is widely recognized as highly sensitive -- it encodes biographic information (date of birth, gender, region) and its misuse enables identity fraud. The JMBG's sensitivity is acknowledged in Serbian legal practice and the Commissioner's public statements.

Bosnia & Herzegovina -- ZZLP BiH (Sl. glasnik BiH 49/2006, updated 2025)

The new ZZLP BiH (published February 28, 2025, entering force March 8, 2025) is harmonized with GDPR. AZLP BiH guidance on technical measures mentions "encryption of data" and "pseudonymization" as recommended measures. No specific mandate for field-level database encryption exists, but the general requirement for "appropriate security measures" applies.

Summary: No Law Mandates Field-Level Encryption

No regulation in any of the three jurisdictions explicitly mandates application-level field-level encryption for tax IDs or IBANs in databases. All three frameworks use GDPR-style language: "appropriate technical measures proportionate to the risk." This means the decision is a **risk-based engineering judgment**, not a binary legal requirement.

1.4 What Competitors Do

Competitor	Approach	Evidence
FreeAgent (UK accounting SaaS, 150K+ users)	AES-256 encryption at rest for all stored data. No public mention of field-level encryption for specific columns. Cyber Essentials Plus certified. AWS Ireland hosting with ISO 27001/27017/27018 datacenter compliance.	FreeAgent Security
Fiken (Norwegian accounting SaaS, 70K+ users)	No publicly available security architecture documentation. Cloud-based, Norwegian-regulated. No evidence of field-level encryption.	General inference from public documentation.
Minimax (Balkan/SEE accounting SaaS)	No publicly available security architecture documentation. Cloud-based. Standard privacy policy. No evidence of field-level encryption for tax identifiers.	Minimax Privacy

Industry pattern: Accounting SaaS competitors rely on disk-level encryption at rest (AES-256, provided by cloud hosting) combined with TLS in transit, access controls, and audit logging. None publicly document field-level encryption for tax IDs or IBANs.

1.5 Risk Assessment by Field

Field	Sensitivity if Breached	Public Availability	Risk Level
JMBG	**CRITICAL** -- enables identity fraud, encodes DOB/gender/region, irrevocable (cannot be changed)	NOT public -- protected by law, not printed on invoices	**Highest**
OIB	**HIGH** -- unique cross-system identifier, used for all gov/financial interactions, difficult to change	Semi-public -- used widely but not freely published	**High**
PIB	**MEDIUM** -- business tax ID, publicly searchable on APR (Serbian Business Registry) and SEF portal	**PUBLIC** -- printed on every invoice, searchable on gov portals	**Medium**
JIB	**MEDIUM** -- business tax ID, publicly searchable on BiH court registry and UIO portal	**PUBLIC** -- printed on every invoice, searchable on gov portals	**Medium**

Field	Sensitivity if Breached	Public Availability	Risk Level
IBAN	**MEDIUM** -- bank account number, shared routinely for payment purposes, printed on invoices	**Semi-public** -- shared with every business partner for payment	**Medium**

1.6 Technical Constraints

Prisma field-level encryption trade-offs:

Factor	Impact
Query limitations	Encrypted fields cannot be filtered, sorted, or searched with SQL operators. Only exact-match via HMAC hash column is possible.
Storage overhead	AES-256-GCM ciphertext is significantly larger than plaintext (IV + auth tag + ciphertext, base64-encoded). VARCHAR columns must be oversized.
Performance	Encrypt/decrypt on every read/write. No published benchmarks for `prisma-field-encryption`, but crypto operations add measurable latency per record. For batch operations (e.g., 1000-invoice report filtering by buyer PIB), overhead compounds.
Key management	Single `FIELD_ENCRYPTION_KEY` in Railway env var. Key rotation requires re-encrypting all rows -- a migration-level operation.
Prisma compatibility	`prisma-field-encryption` library uses Prisma client extensions (AES-256-GCM). Works but adds a dependency. Raw SQL queries (used for financial reports per ADR-011) bypass encryption middleware.
Development complexity	Developers must handle encrypted fields differently. Debugging queries on encrypted columns is harder. Test fixtures need encryption-aware setup.

---

2. Decision

We will use a hybrid approach:

Tier 1: Field-Level Encryption (AES-256-GCM) -- JMBG and OIB only

These are personal identification numbers with high breach impact and no legitimate reason for database-level querying by value:

- **JMBG** (Serbia/BiH citizen number): Encrypted at application layer before storage. HMAC-SHA256 hash stored in `jmbgHash` column for exact-match lookup when required.
- **OIB** (Croatia personal/company ID): Encrypted at application layer before storage. HMAC-SHA256 hash stored in `oibHash` column for exact-match lookup.

Implementation: Use `prisma-field-encryption` Prisma client extension with `/// @encryption:encrypt` annotation on JMBG and OIB fields in schema.prisma. Add `/// @encryption:hash(jmbg)` for searchable hash columns.

Rationale: JMBG and OIB are irrevocable personal identifiers. A database breach exposing plaintext JMBG enables identity fraud with no mitigation path for the victim (JMBG cannot be changed). The risk justifies the query limitations and performance overhead because:

- These fields are rarely queried in bulk (lookup is by known value, not range/partial match)
- These fields are displayed to users infrequently (only on contact detail views, not list views)
- The DPIA (section 3) already documents AES-256-GCM for these fields -- implementing it fulfills the documented risk mitigation

Tier 2: Disk-Level Encryption + Application Controls -- PIB, JIB, IBAN

These are business identifiers or semi-public financial data where the risk profile does not justify the significant query/performance trade-offs of field-level encryption:

- **PIB** (Serbia tax ID): Publicly searchable on APR. Printed on every invoice. Encrypted at disk level (Railway AES-256). Protected by org-scoping middleware, RBAC, audit trail, and TLS in transit.
- **JIB** (BiH tax ID): Publicly searchable on court registry. Printed on every invoice. Same controls as PIB.
- **IBAN**: Shared with every business partner for payment. Printed on invoices. Masked in list responses (show last 4 digits only). Same disk-level + application-layer controls.

Rationale: Encrypting PIB at the field level when it is publicly available on Serbian government portals (APR, efaktura.mfin.gov.rs) provides negligible additional security benefit while significantly degrading query performance. The same applies to JIB. IBAN is routinely shared for payment and is masked in API list responses. For all three, the defense-in-depth stack (disk encryption + TLS + org-scoping + RBAC + audit trail + API masking) provides security proportionate to the risk, consistent with GDPR Article 32's risk-based approach.

Implementation Summary

Field	Encryption Level	Search Support	Display Masking
JMBG	AES-256-GCM field-level	HMAC-SHA256 hash for exact match	Show only last 3 digits in UI
OIB	AES-256-GCM field-level	HMAC-SHA256 hash for exact match	Show only last 3 digits in UI
PIB	Disk-level (Railway AES-256)	Full SQL query support	Full value shown (public data)
JIB	Disk-level (Railway AES-256)	Full SQL query support	Full value shown (public data)
IBAN	Disk-level (Railway AES-256)	Full SQL query support	Masked in list views (last 4 only), full in detail view

Key Management

- `FIELD_ENCRYPTION_KEY`: 32-byte hex string stored in Railway secrets. Used for JMBG and OIB encryption only.
- `FIELD_HMAC_KEY`: Separate 32-byte hex string stored in Railway secrets. Used for deterministic hash columns.
- Key rotation: Annual. Rotation requires a migration script to re-encrypt all JMBG/OIB rows -- scoped to Contact table only (manageable volume at MVP scale).
- Per Key Management Policy (POL-SEC-KM-001), keys are never committed to source code and are provisioned only through Railway environment secrets or Vaultwarden.

---

3. Alternatives Considered

Option A: No field-level encryption for any L4 field (original Security Architecture position)

Pros:

- Zero query limitations -- PIB, JMBG, OIB, JIB, IBAN all fully searchable/sortable
- Zero performance overhead from application-layer crypto
- Simpler development -- no encryption middleware, no hash columns, no key management
- Consistent with what competitors do (FreeAgent, Fiken, Minimax)
- Disk-level encryption (Railway AES-256) protects against physical theft and disk-level breaches

Cons:

- A database breach (e.g., SQL injection bypassing Prisma, Railway compromise, backup leak) exposes all JMBG/OIB in plaintext
- JMBG exposure is irrevocable -- victims cannot change their JMBG
- Contradicts the DPIA (section 3) which documents field-level encryption as a committed mitigation
- Weaker posture for regulatory audits and GDPR accountability demonstrations
- Negligent if a breach occurs and DPA asks why field-level encryption was not implemented despite being documented in the DPIA

Why not chosen: The risk of JMBG/OIB exposure in a breach is too high to leave unmitigated. Removing a documented DPIA control without justification creates regulatory liability.

Option B: AES-256-GCM field-level encryption for ALL L4 fields (Encryption Policy position) -- Selected with modification

Pros:

- Maximum defense-in-depth -- all L4 data encrypted at field level
- Simplest compliance narrative ("all sensitive data is encrypted at rest at the field level")
- Matches the DPIA and Encryption Policy as written

Cons:

- PIB and JIB are publicly available data -- encrypting them provides marginal security benefit
- IBAN is shared routinely -- encrypting it adds complexity for minimal breach-impact reduction
- Encrypting PIB/JIB breaks SQL `WHERE` clause filtering for invoice lookups by buyer tax ID -- a common query pattern for SEF e-invoice reconciliation and accounting workflows
- Encrypting IBAN breaks bank reconciliation queries
- Performance overhead on every invoice read/write for fields that are public or semi-public
- Over-engineering -- GDPR Art. 32 requires measures "appropriate to the risk," not maximum possible measures
- No competitor in the accounting SaaS space implements field-level encryption for business tax IDs

Why not chosen in full: Encrypting publicly available identifiers (PIB, JIB) at the field level fails the GDPR Art. 32 proportionality test. The cost (query limitations, performance, complexity) is not justified by the risk reduction (minimal, since the data is publicly accessible). However, the principle of encrypting personal identifiers (JMBG, OIB) is correct and adopted.

Option C: Hybrid approach -- Encrypt JMBG and OIB only (Selected)

Pros:

- Proportionate to risk: highest-sensitivity fields get strongest protection
- Preserves full query capability for business tax IDs (PIB, JIB) and IBAN -- critical for accounting workflows
- Manageable scope: only Contact table fields encrypted, not Organization or BankAccount
- HMAC hash columns enable exact-match lookup for the encrypted fields
- Consistent with GDPR Art. 32 risk-based approach
- Defensible in regulatory audit: "we assessed risk per field and applied measures proportionate to each"
- Limited performance impact: JMBG/OIB are read infrequently (contact detail views only)

Cons:

- More nuanced compliance narrative than "everything is encrypted"
- Two tiers of L4 data treatment requires clear documentation
- IBAN remains in plaintext at database level (mitigated by disk encryption + masking)

Risk accepted: A database breach would expose PIB, JIB, and IBAN in plaintext within the encrypted disk. This is accepted because: (a) PIB and JIB are publicly available, (b) IBAN is routinely shared and masked in API responses, (c) disk-level encryption protects against physical/infrastructure breaches, (d) org-scoping prevents cross-tenant access at application level.

Option D: PostgreSQL pgcrypto column-level encryption (database-side)

Pros:

- Encryption happens at the database layer -- transparent to application code
- Works with raw SQL queries (unlike Prisma middleware)

Cons:

- Requires `pgp_sym_encrypt()` / `pgp_sym_decrypt()` in every query -- significantly increases SQL complexity
- Key must be passed in every query or stored in a PostgreSQL session variable -- key management is harder
- Railway managed PostgreSQL may not support pgcrypto extension
- Prisma has no native support for pgcrypto -- requires `$queryRaw` for all encrypted field operations, negating the benefits of the Prisma ORM (ADR-011)
- Breaks Prisma schema-as-code principle

Why not chosen: Incompatible with Prisma ORM and Railway managed PostgreSQL constraints. Application-layer encryption via `prisma-field-encryption` is a better fit for the existing stack.

---

4. Consequences

4.1 Positive Consequences

- JMBG and OIB are protected against database breaches -- even full DB dump yields only ciphertext
- HMAC hash columns enable exact-match lookup without exposing plaintext
- PIB, JIB, IBAN remain fully queryable -- no impact on accounting workflows, SEF reconciliation, or bank reconciliation
- Risk-proportionate approach is defensible under GDPR Art. 32 and ZZPL Art. 50
- Aligns with DPIA documented mitigations (AES-256-GCM for personal identifiers)
- Key management scope is limited (one encryption key, one HMAC key, scoped to two field types)

4.2 Negative Consequences

- JMBG and OIB fields cannot be filtered, sorted, or partially matched in SQL -- only exact-match via hash -- *Mitigation: These fields are rarely queried in bulk; exact-match via HMAC covers the primary use case (lookup by known value)*
- `prisma-field-encryption` library is a third-party dependency (47ng/prisma-field-encryption, MIT license) -- *Mitigation: Library is well-maintained, 600+ GitHub stars, uses Node.js native `crypto` module internally; can be replaced with custom Prisma extension if abandoned*
- Key rotation requires re-encryption migration -- *Mitigation: Scoped to Contact table rows only; at MVP scale (<10K contacts), migration completes in seconds*
- Raw SQL queries (used for financial reports per ADR-011) bypass Prisma encryption middleware -- *Mitigation: JMBG/OIB are never included in financial report queries; they are contact metadata, not transaction data*

4.3 Technical Debt Created

- DATA-ENCRYPTION-POLICY.md and COMPLIANCE-FRAMEWORK.md must be updated to reflect the tiered L4 approach (JMBG/OIB = field-level, PIB/JIB/IBAN = disk-level + controls) instead of blanket "all L4 = field-level" -- *Plan: Update immediately after ADR approval*
- SECURITY-ARCHITECTURE.md line "Column-level encryption: Not needed" must be updated to reflect the hybrid decision -- *Plan: Update immediately after ADR approval*
- DPIA data inventory table should differentiate encryption tier per field -- *Plan: Update in next DPIA revision*

4.4 Schema Changes Required

```
model Contact {
  // ... existing fields ...

  // Tier 1: Field-level encrypted (AES-256-GCM)
  jmbg       String?  @db.Text  /// @encryption:encrypt  -- Serbian/BiH citizen number
  jmbgHash   String?  @map("jmbg_hash") @db.VarChar(64)  /// @encryption:hash(jmbg)
  oib        String?  @db.Text  /// @encryption:encrypt  -- Croatian personal ID
  oibHash    String?  @map("oib_hash") @db.VarChar(64)   /// @encryption:hash(oib)

  // Tier 2: Disk-level encryption only (full query support)
  // registrationNumber (PIB/JIB) -- already exists, no change
  // vatNumber -- already exists, no change
}

model BankAccount {
  // iban field -- already exists as @db.VarChar(50), no encryption change
  // Protected by: disk encryption + org-scoping + RBAC + API masking
}
```

4.5 Environment Variables Required

```
# Generate with: openssl rand -hex 32
FIELD_ENCRYPTION_KEY=<64-char-hex-string>  # AES-256 key for JMBG/OIB encryption
FIELD_HMAC_KEY=<64-char-hex-string>         # HMAC-SHA256 key for hash columns
```

---

5. Decision Rationale Summary

The core insight driving this decision is that **not all L4 data carries equal breach risk**. The original Security Architecture was too permissive (no field-level encryption at all). The Encryption Policy was too aggressive (field-level encryption for publicly available business IDs). This ADR resolves the conflict with a risk-proportionate hybrid:

1. **JMBG/OIB**: Irrevocable personal identifiers with high breach impact. Field-level encryption is justified and proportionate.
2. **PIB/JIB**: Publicly available business identifiers. Field-level encryption adds cost without meaningful risk reduction.

3. **IBAN**: Routinely shared financial identifier. API masking + disk encryption is proportionate.

This approach satisfies GDPR Article 32's "appropriate to the risk" standard, avoids the EDPB-documented pitfall of "security theatre" (implementing controls that do not meaningfully reduce risk), and preserves the query performance essential for accounting workflows.

References

- GDPR Article 32 -- Security of Processing
- GDPR Article 87 -- Processing of National Identification Numbers
- Serbia ZZPL (Sl. glasnik RS 87/2018)
- BiH ZZLP -- AZLP Security Measures Guidance
- Croatia AZOP -- Data Protection Overview
- prisma-field-encryption Library
- FreeAgent Security
- ICO Encryption Guidance
- JMBG Structure and Sensitivity
- Bilko DPIA (docs/security/DPIA.md)
- Bilko Data Encryption Policy (docs/security/DATA-ENCRYPTION-POLICY.md)
- Bilko Compliance Framework (docs/security/COMPLIANCE-FRAMEWORK.md)
- Bilko Key Management Policy (docs/security/KEY-MANAGEMENT-POLICY.md)

Approval

Role	Name	Date	Signature
Author	Petter Graff	2026-02-25	
Tech Lead			
DPO			
CTO / Architect	Alem		

ADR-015: Four-Jurisdiction Plugin Architecture

```
# ADR-015 — Four-Jurisdiction Plugin Architecture (CountryPlugin Kotlin Interface)

**Status:** Accepted
**Date:** 2026-05-13
**Author:** Petter Graff (CodeCraft — Architecture Lead)
**Decision-maker:** CEO Alem Baši?
**MC Task:** #100585 (Phase 0' ADR Consolidation — CountryPlugin interface)
**Supersedes:** ADR-015 v1 (2026-05-11, MC #100362) — this is the authoritative version
**Cross-references:**

- ADR-016 (EInvoiceAdapter — `generateEInvoiceXml()` and `submitToFiscalPlatform()`
delegate to it)
- ADR-017 (RLS multi-tenancy — `TaxJurisdiction` enum drives `country_code` column values)
- ADR-019 (Integration Adapter Registry — adapters called by plugin implementations)
- ADR-023 (transitional routing — single backend, market selected from org record)
- ADR-bilko-001 (promoted as ADR-017 — Option C single-DB decision context)
- ADR-bilko-002 (extraction strategy — Variant C package isolation rationale)
- ADR-bilko-003 (3-layer market abstraction — CountryPlugin is Layer 1)
- Plan v3 §4a, §4b, §5, §6 Phase 0' — `~/system/specs/bilko-multi-market-architecture-plan-
v3-2026-05-11.md`

---

## 1. Context

### 1.1 Current State (tool-verified 2026-05-11)

The Kotlin/Ktor backend (`bilko-api-demo`, Cloud Run, europe-north1) serves three brand
hostnames (bilko.cloud, bilko.company, bilko.io) via a single runtime. ADR-023 established
this as the deliberate transitional architecture. Market differentiation is currently
handled
by two mechanisms:

1. `ComplianceCalendarService.kt` — manual `when(organization.country)` branching
2. `StorecoveHrFiskEInvoiceAdapter.kt` — directly implements `EInvoiceAdapter`; no
   `CountryPlugin` wrapper exists

Neither mechanism is pluggable. Adding a fourth market requires editing shared service
files.
This violates the Open/Closed principle and creates unbounded audit surface.

**Verified absence:** `find ... -name "CountryPlugin.kt"` returns zero results (v3 plan
§2).
`TaxJurisdiction.kt` currently has: `HR, RS, BA` (BA conflates two distinct fiscal
jurisdictions).

**JWT reality (tool-verified):** `JwtService.kt` embeds `orgId` in the JWT, NOT
`org.country`.
The `org.country` value is fetched from the `organizations` DB table via `orgId` in the
request
middleware. All DI wiring in this ADR reflects this two-step lookup.

### 1.2 Problem

Without a plugin abstraction:

- Each new market forces edits to `ComplianceCalendarService`, `InvoiceService`, and any
  other service that branches on `organization.country`.
- The Open/Closed principle is violated: adding Bosnia FBiH requires modifying existing
  code across multiple files, not extending it.
- Tax auditors reviewing Croatian PDV compliance must read shared files that also contain
  Serbian PDV logic — audit surface is unbounded.
- `StorecoveHrFiskEInvoiceAdapter` has no dispatch mechanism routing "HR org, generate
```

  invoice" to it cleanly.

1.3 BA Split Rationale

Bosnia-Herzegovina is not a single fiscal jurisdiction:

Dimension	BA-FED	BA-RS
Formal name	Federacija BiH	Republika Srpska entity
Tax authority	UIO-FBiH (Uprava za indirektno oporezivanje FBiH)	Poreska Uprava RS entity
E-invoice platform	CPF (stub, mandatory ~2027)	UINO (stub, mandatory TBD)
Filing pravilnik	FBiH Pravilnik o kontnom okviru	RS entity Pravilnik
Company identifier	JIB (13 digits)	JIB (13 digits)
PDV rate	17% standard, no reduced	17% standard, no reduced
Currency	BAM	BAM

A single `PluginBA` with internal branching reproduces the Variant B coupling problem (ADR-bilko-002 §3). The split is required.

---

2. Decision

2.1 TaxJurisdiction Enum — Canonical Form

```kotlin
package no.alai.bilko.country

/**
 * Canonical tax jurisdictions supported by Bilko.
 *
 * DB column constraint: CHECK country_code IN ('HR', 'RS', 'BA_FED', 'BA_RS')
 * NOTE: BA bare value is retained in the Kotlin enum during the V16 migration window
 * to allow backfill of existing DB rows. Remove BA after V16 validates on prod.
 *
 * See: ADR-015 §2.1, Plan v3 §6 Phase 1H.1
 */
enum class TaxJurisdiction {
    HR,     // Croatia — EUR, Storecove/Peppol via FINA AS4, PDV 25/13/5%
    RS,     // Serbia — RSD, SEF (Sistem e-faktura), PDV 20/10%
    BA,     // Bosnia bare value — DEPRECATED, retained for V16 backfill window only
    BA_FED, // Bosnia FBiH — BAM, CPF e-invoice (stub), UIO-FBiH, FBiH Pravilnik, PDV 17%
    BA_RS,  // Bosnia RS entity — BAM, UINO (stub), Poreska Uprava RS entity, PDV 17%
}
```

Migration note: Flyway V16 backfills `BA ? BA_FED` rows, then adds the NOT NULL + CHECK constraint. `BA` is removed from the enum in a cleanup MC after V16 validates on prod.

2.2 CountryPlugin Interface — Full Contract

Written to `apps/api/src/main/kotlin/no/alai/bilko/country/CountryPlugin.kt`.

Interface invariant: Zero `if jurisdiction ==` branches in core services (`services/`, `routes/`). All market differences are absorbed here.

```kotlin
package no.alai.bilko.country

import no.alai.bilko.einvoice.CanonicalInvoice
import no.alai.bilko.einvoice.EInvoiceAdapter
import java.util.Currency

/**
 * Per-jurisdiction plugin — single extension point for all market-specific behaviour.
 *
```

```
 * INVARIANT: No `if jurisdiction == X` or `when(jurisdiction)` branches in
 * apps/api/src/main/kotlin/no/alai/bilko/{services,routes}/.
 * All market differences are absorbed here. (ADR-bilko-002 Variant C)
 *
 * Implementations:
 *   PluginHR      ? country/hr/PluginHR.kt          (Phase 1H — priority)
 *   PluginRS      ? country/rs/PluginRS.kt           (stub, Phase 1S)
 *   PluginBAFED   ? country/ba/PluginBAFED.kt        (stub, Phase 1B)
 *   PluginBARS    ? country/ba/PluginBARS.kt         (stub, Phase 1B)
 *
 * DI: plugins/DI.kt registers all 4 in a Map<TaxJurisdiction, CountryPlugin>.
 * Resolution: orgId from JWT ? DB lookup organizations.country ? TaxJurisdiction.valueOf()
 * ? PluginRegistry.resolve() (see ADR-015 §2.4 for full pipeline).
 */
interface CountryPlugin {

    /**
     * Returns the tax jurisdiction this plugin handles.
     * Used by PluginRegistry to route. Must be consistent with the plugin's
     * registration key in the DI map.
     */
    fun jurisdiction(): TaxJurisdiction

    /**
     * Calculates VAT breakdown for the given canonical invoice.
     *
     * Returns [VatResult] containing itemised tax lines per rate band.
     * Core invoice service calls this; NEVER inspects jurisdiction directly.
     *
     * HR: 25% (S — standard), 13% (AA — reduced-1), 5% (E — reduced-2), 0% (Z —
zero/export)
     * RS: 20% (standard), 10% (reduced), 0% (export)
     * BA-FED / BA-RS: 17% (standard), 0% (export)
     *
     * @throws UnsupportedOperationException for stub implementations (RS, BA)
     */
    fun calculateVat(invoice: CanonicalInvoice): VatResult

    /**
     * Generates jurisdiction-specific e-invoice bytes from the canonical model.
     *
     * Delegates to the platform-specific [EInvoiceAdapter.serialize()] for this
jurisdiction.
     * Returns the wire-format payload (UBL 2.1 XML Storecove envelope for HR; SEF XML for
RS).
     * Contract: OFFLINE — no network, no credentials required.
     *
     * @throws UnsupportedOperationException for stub implementations
     */
    fun generateEInvoiceXml(invoice: CanonicalInvoice): ByteArray

    /**
     * Submits a previously serialized e-invoice to the fiscal platform.
     *
     * [receipt] bundles the serialized bytes from [generateEInvoiceXml] with the
originating
     * [CanonicalInvoice] for idempotency key generation.
     * Returns [FiscalSubmissionHandle] with the platform submission ID.
     * Throws [no.alai.bilko.adapter.AdapterException] on all failure modes.
     *
     * HR lifecycle: STUB until MC #8675 (Storecove account activation).
     */
    fun submitToFiscalPlatform(receipt: FiscalReceipt): FiscalSubmissionHandle

    /**
     * Returns default Chart of Accounts entries for this jurisdiction.
     *
     * Called once on org creation to seed the tenant's account list with the
     * mandatory Pravilnik accounts. Company may add or rename — these are minimums.
     *
     * HR: FINA Kontni Plan (11-year retention)
     * RS: Serbian Pravilnik (10-year retention)
     * BA: FBiH / RS entity Pravilnik (10-year retention)
     */
    fun getChartOfAccountsDefaults(): List<ChartOfAccountEntry>
```

```
    /**
     * Returns filing deadline schedule for this jurisdiction.
     *
     * Returns a sorted list of [FilingDeadline] for the next 12 months from the call date.
     * Used by ComplianceCalendarService to populate per-org reminder schedules.
     *
     * HR: quarterly PDV return (last working day of month after quarter end) + annual CIT
(30 April)
     * RS: monthly PDV return (within 15 days of month end)
     * BA: FBiH / RS entity PDV return schedules
     */
    fun getFilingDeadlines(): List<FilingDeadline>

    /**
     * Returns data retention policy for this jurisdiction.
     *
     * HR: 11 years — Zakon o ra?unovodstvu NN 78/2015, ?l. 10
     * RS: 10 years — Zakon o ra?unovodstvu RS
     * BA-FED / BA-RS: 10 years
     *
     * Used by the document archiving service to set per-org retention periods and by
     * the RLS audit partition (ADR-017 Phase 2B).
     */
    fun getRetentionRules(): RetentionPolicy

    /**
     * Returns the functional currency for this jurisdiction.
     *
     * HR: Currency.getInstance("EUR")  — Croatia adopted EUR 2023-01-01
     * RS: Currency.getInstance("RSD")
     * BA-FED / BA-RS: Currency.getInstance("BAM")
     *
     * Core invoice service validates CanonicalInvoice.currencyCode against this on
creation.
     */
    fun getCurrency(): Currency

    /**
     * Returns locale-specific formatters for this jurisdiction.
     *
     * HR: decimal='.', thousands=',', date='dd.MM.yyyy', tz='Europe/Zagreb'
     * RS: decimal=',', thousands='.', date='dd.MM.yyyy', tz='Europe/Belgrade'
     * BA: decimal=',', thousands='.', date='dd.MM.yyyy', tz='Europe/Sarajevo'
     *
     * Used by report generation, PDF invoices, and UI date/number display.
     */
    fun getFormatters(): JurisdictionFormatters

    /**
     * Extension hook for jurisdiction-specific validation beyond the standard 8 methods.
     *
     * Called by InvoiceService before invoice creation. Default implementation is a no-op;
     * override to add market-specific business rules (e.g., HR OIB cross-validation
     * against FINA company registry once APRCompanyRegistryAdapter is live).
     *
     * This hook is the designated extension point to avoid adding new required interface
     * methods for market-specific edge cases. See §3.2 for evolution contract.
     *
     * @param invoice draft canonical invoice before persistence
     * @throws no.alai.bilko.adapter.AdapterException with VALIDATION_BUSINESS_RULE if
invalid
     */
    fun validateInvoiceForJurisdiction(invoice: CanonicalInvoice) {
        // Default: no-op. Override in PluginHR, PluginRS etc. as needed.
    }
}
```

2.3 Supporting Value Types

Defined in `no.alai.bilko.country` package (or `no.alai.bilko.country.model`):

```kotlin
// VAT calculation result
```

```
data class VatResult(
    val lines: List<VatLine>,
    val totalVatAmount: java.math.BigDecimal,
    val totalTaxableAmount: java.math.BigDecimal,
)

data class VatLine(
    val rate: java.math.BigDecimal,          // e.g. BigDecimal("25.0000")
    val category: no.alai.bilko.einvoice.TaxCategory,
    val taxableAmount: java.math.BigDecimal,
    val taxAmount: java.math.BigDecimal,
    val description: String,                 // Human-readable, e.g. "HR standard PDV 25%"
)

// Fiscal submission input
data class FiscalReceipt(
    val serializedInvoice: ByteArray,
    val canonicalInvoice: no.alai.bilko.einvoice.CanonicalInvoice,
)

data class FiscalSubmissionHandle(
    val platformInvoiceId: String,           // Storecove GUID, SEF ID, etc.
    val initialStatus: no.alai.bilko.einvoice.EInvoiceStatus,
    val submittedAt: java.time.Instant,
)

// Chart of Accounts entry
data class ChartOfAccountEntry(
    val code: String,                        // e.g. "1300" (HR) or "204" (RS)
    val name: String,
    val type: AccountType,                   // ASSET, LIABILITY, EQUITY, INCOME, EXPENSE
    val vatTreatment: String?,
)

// Filing deadline
data class FilingDeadline(
    val name: String,                        // e.g. "Quarterly PDV return Q1 2026"
    val dueDate: java.time.LocalDate,
    val authority: String,                   // e.g. "Porezna uprava HR (ePorezna)"
    val periodStart: java.time.LocalDate,
    val periodEnd: java.time.LocalDate,
)

// Data retention
data class RetentionPolicy(
    val years: Int,                          // 10 or 11 depending on jurisdiction
    val legalBasis: String,                  // Statutory reference
    val jurisdiction: TaxJurisdiction,
)

// Formatters
data class JurisdictionFormatters(
    val decimalSeparator: Char,
    val thousandsSeparator: Char,
    val datePattern: String,                 // ISO strftime-compatible, e.g. "dd.MM.yyyy"
    val timeZoneId: String,                  // IANA tz, e.g. "Europe/Zagreb"
    val currencySymbol: String,
    val currencyPosition: CurrencyPosition,  // PREFIX or SUFFIX
)

enum class CurrencyPosition { PREFIX, SUFFIX }
```

2.4 DI Wiring Strategy

JWT reality: The JWT access token contains `orgId` only (verified in `JwtService.kt` lines 35–45). The `org.country` value is NOT embedded in the JWT. It is fetched from the `organizations` DB table at request time by middleware before the route handler runs.

Resolution pipeline:

```
HTTP request
    ? JWT validation (JwtService.verifyAccessToken)
    ? extract orgId from JWT claim "orgId"
```

```
    ? DB: SELECT country FROM organizations WHERE id = orgId  (OrgScopePlugin / middleware)
    ? TaxJurisdiction.valueOf(country)
    ? PluginRegistry.resolve(jurisdiction)
    ? CountryPlugin dispatch
```

DI registration in `plugins/DI.kt`:

```kotlin
// Phase 1H Task 1H.4
val pluginRegistry: Map<TaxJurisdiction, CountryPlugin> = mapOf(
    TaxJurisdiction.HR     to PluginHR(StorecoveHrFiskEInvoiceAdapter()),
    TaxJurisdiction.RS     to PluginRS(),      // stub — Phase 1S
    TaxJurisdiction.BA_FED to PluginBAFED(),   // stub — Phase 1B
    TaxJurisdiction.BA_RS  to PluginBARS(),    // stub — Phase 1B
)

// In Koin module:
single<Map<TaxJurisdiction, CountryPlugin>> { pluginRegistry }

// Resolution helper (usable from any Koin-injected service):
fun resolvePlugin(
    jurisdiction: TaxJurisdiction,
    registry: Map<TaxJurisdiction, CountryPlugin>
): CountryPlugin = registry[jurisdiction]
    ?: throw IllegalStateException(
        "No CountryPlugin registered for $jurisdiction — check DI.kt registration"
    )
```

Services that need a `CountryPlugin` receive it via constructor injection:

```kotlin
class InvoiceService(
    private val pluginRegistry: Map<TaxJurisdiction, CountryPlugin>
    // ... other deps
) {
    private fun plugin(org: Organization): CountryPlugin =
        resolvePlugin(TaxJurisdiction.valueOf(org.country), pluginRegistry)
}
```

2.5 OrgScopePlugin Sequencing Decision

Decision: CountryPlugin resolution runs AFTER OrgScopePlugin (org isolation middleware).

Rationale:

1. **Security gate must run first.** OrgScopePlugin validates that the authenticated user belongs to the org being operated on and sets the `app.current_org_id` Postgres session variable for RLS PERMISSIVE enforcement (Phase 2A). This is a security boundary; no business logic should execute before it.

2. **CountryPlugin requires an authenticated, org-scoped context.** Resolving a `CountryPlugin` requires reading `organizations.country` from DB, which in turn requires a verified `orgId`. OrgScopePlugin is what establishes and validates that `orgId`.

3. **Failure mode is clean.** If OrgScopePlugin fails (user not in org, org not found), the request is rejected with 403 before CountryPlugin resolution is attempted. No country-specific logic runs on unauthenticated requests.

Execution order in the Ktor pipeline:

```
1. Authentication plugin (JWT validation)
2. OrgScopePlugin:
   a. Validate user.org_id matches the resource being accessed
   b. SET app.current_org_id = :orgId  (for RLS)
   c. Fetch org record ? populate OrgContext (includes org.country)
3. CountryPlugin resolution:
   a. TaxJurisdiction.valueOf(orgContext.country)
   b. resolvePlugin(jurisdiction) ? inject into route handler
4. Route handler executes with both OrgContext and CountryPlugin available
```

Parisa Tabriz (Securion) note: OrgScopePlugin must complete step 2b before any CountryPlugin method is called. This ensures the RLS session variable is set before any DB query inside the plugin executes. Violating this order creates a window where a CountryPlugin DB query runs without the RLS filter active.

2.6 TypeScript Packages — Separate Concern

The five TypeScript packages (`packages/domain-rs`, `packages/domain-hr`, `packages/domain-ba`,
`packages/domain-ba-fed`, `packages/domain-ba-rs`) contain frontend domain types compiled to
`dist/`. They are **not loaded by the Kotlin runtime** and are **not in scope for this ADR**.

The `TaxJurisdiction` enum values must remain consistent between the Kotlin enum and any TypeScript enums in these packages (same string values: `"HR"`, `"RS"`, `"BA_FED"`, `"BA_RS"`).
That alignment is enforced at the API boundary (JWT claim and REST API JSON) — not via a shared runtime dependency.

Backwards compatibility rule: if `TaxJurisdiction` gains a new value (e.g., `SI` for Slovenia),
the corresponding TypeScript packages must be updated in the same PR. This is a documentation
constraint, not a compile-time enforcement.

---

3. Enforcement

3.1 Linting Rule

A custom Detekt rule must reject any file in `apps/api/src/main/kotlin/no/alai/bilko/{services,routes}/` that contains patterns:

- `if.*jurisdiction`
- `when.*jurisdiction`
- `if.*country ==`
- `when.*country`

This rule is a Phase 1H CI gate. It runs before any Phase 1H code merges to main.
The rule is not applied to `country/` package itself (plugin implementations may internally branch on jurisdiction during their own construction if absolutely necessary).

3.2 Interface Evolution Contract

When a new method must be added to `CountryPlugin`:

1. **Prefer the extension hook** (`validateInvoiceForJurisdiction`) for market-specific validation that does not generalise across all markets.
2. If a new method is genuinely cross-market: add it with a default body that throws `UnsupportedOperationException("Not implemented for $jurisdiction — see MC #XXXX")`.
3. Override in `PluginHR` (priority market) first; other plugins follow in their phase.
4. Default throws surface as clear runtime errors, not silent wrong behaviour.

---

4. Implementation Path

Phase	Task	Status	Files
Phase 0'	This ADR	DONE	`docs/architecture/ADR-015-...md`
Phase 1H.1	`TaxJurisdiction` expanded `{HR,RS,BA,BA_FED,BA_RS}`	Blocked by 0'	`TaxJurisdiction.kt`
Phase 1H.1	`CountryPlugin.kt` interface + supporting types written	Blocked by 0'	`country/CountryPlugin.kt` (NEW)
Phase 1H.2	`PluginHR` implemented (9 methods + hook)	Blocked by 1H.1	`country/hr/PluginHR.kt` (NEW)
Phase 1H.3	`PluginRS`, `PluginBAFED`, `PluginBARS` stubs	Blocked by 1H.1	`country/{rs,ba}/Plugin*.kt`
Phase 1H.4	DI registration; OrgScopePlugin order enforced		

`plugins/DI.kt` | Blocked by 1H.2+3 |
| Phase 1H.5 | Flyway V16 — backfill BA?BA_FED, add NOT NULL + CHECK |
`V16__country_jurisdiction_constraint.sql` | Blocked by 0'3 |
| Phase 1S | `PluginRS` fully implemented |
`country/rs/PluginRS.kt` | Post-HR GA |
| Phase 1B | `PluginBAFED`, `PluginBARS` implemented |
`country/ba/Plugin*.kt` | Post-RS GA |

---

5. Consequences

5.1 Positive

- **Fifth market = one new file.** Adding Slovenia (SI) requires `PluginSI.kt`, one DI registration, and `SI` added to `TaxJurisdiction`. Zero core service changes.
- **Bounded audit surface.** Croatian PDV auditors read `country/hr/PluginHR.kt` only.
- **Team parallelism.** HR sprint and RS sprint work concurrently on separate files.
- **Versioned CoA.** `getChartOfAccountsDefaults()` seeds Pravilnik data; rate changes handled via the versioned `chart_of_accounts` table (ADR-017 §2.4).

5.2 Negative

- **New required method touches all 4 implementations.** Mitigation: default throw pattern (§3.2) + extension hook for non-cross-cutting additions.
- **Boilerplate at scaffolding time.** Each market: ~9 method bodies, CoA seed data, test harness. Estimate: 2 days per market for the core plugin scaffold.
- **OrgScopePlugin coupling.** CountryPlugin resolution depends on OrgScopePlugin having run and fetched the org record. If OrgScopePlugin is ever refactored, the CountryPlugin resolution pipeline must be updated in lockstep.

5.3 Risks

- **Jurisdiction if-branches in core services.** Deadline pressure leads to `if (jurisdiction == TaxJurisdiction.HR)` shortcuts. **Mitigation:** Detekt rule (§3.1).
- **Stub plugin HTTP 500.** If `PluginRS` is a stub and an RS user triggers `calculateVat()`, `UnsupportedOperationException` propagates as HTTP 500. **Mitigation:** DI registry should check `lifecycleState` at request time and return HTTP 503 (market feature not available).
- **BA backfill assumption.** V16 migrates `BA ? BA_FED` as default. If any existing BA org is actually RS entity, the assumption is wrong. **Mitigation:** CEO notified before V16 runs on prod; manual verification of all BA rows (currently 0 paying customers).

---

6. References

Reference	Path	Lines Referenced
`TaxJurisdiction.kt` (current)	`apps/api/src/main/kotlin/no/alai/bilko/country/TaxJurisdiction.kt`	1–23
`JwtService.kt` (JWT claims — orgId only)	`apps/api/src/main/kotlin/no/alai/bilko/auth/JwtService.kt`	35–45
`BilkoPrincipal.kt`	`apps/api/src/main/kotlin/no/alai/bilko/auth/BilkoPrincipal.kt`	1–10
`EInvoiceAdapter` interface	`apps/api/src/main/kotlin/no/alai/bilko/einvoice/EInvoiceTypes.kt`	200–224
`StorecoveHrFiskEInvoiceAdapter.kt` (HR reference)	`apps/api/src/main/kotlin/no/alai/bilko/country/hr/StorecoveHrFiskEInvoiceAdapter.kt`	537–777
`DI.kt` (current Koin module — no country plugin yet)	`apps/api/src/main/kotlin/no/alai/bilko/plugins/DI.kt`	1–67
Plan v3 §2 current state truth	`~/system/specs/bilko-multi-market-architecture-plan-v3-2026-05-11.md`	28–73

Plan v3 §4a (Option D not triggered)	`~/system/specs/bilko-multi-market-architecture-plan-v3-2026-05-11.md`	100–119
Plan v3 §4b (Phase 0 ADR scope)	`~/system/specs/bilko-multi-market-architecture-plan-v3-2026-05-11.md`	121–133
Plan v3 §6 Phase 0' Task 0'1	`~/system/specs/bilko-multi-market-architecture-plan-v3-2026-05-11.md`	246–255

---

7. Approval

Status: Accepted — no CEO sign required (architecture contract, not data migration)

Unblocks:

- Phase 1H Task 1H.1: `TaxJurisdiction` enum expansion + `CountryPlugin.kt`
- Phase 1H Task 1H.2: `PluginHR` implementation
- ADR-016: EInvoiceAdapter contract (referenced from `generateEInvoiceXml()`)
- ADR-019: Adapter Registry (referenced from `submitToFiscalPlatform()`)

Role	Sign	Date
Architecture Lead (Petter Graff)	Signed	2026-05-13
CEO (Alem Baši?)	Not required for interface ADR	—

---

8. Document History

Date	Author	Change
2026-05-11	Petter Graff	v1 — Phase 0' initial (MC #100362)
2026-05-13	Petter Graff	v2 — MC #100585: OrgScopePlugin sequencing decision; JWT reality (orgId, not country claim); extension hook `validateInvoiceForJurisdiction`; TypeScript packages backwards-compat section; DI wiring corrected to reflect actual JwtService contract

ADR-016: EInvoice Adapter Lifecycle and Contract

```
# ADR-016 — EInvoiceAdapter Lifecycle and Contract

**Status:** Accepted
**Date:** 2026-05-13
**Author:** Petter Graff (CodeCraft — Architecture Lead)
**Finverge Co-author:** Markos Zachariadis (Payments & Fiscal Integration)
**Decision-maker:** CEO Alem Baši?
**MC Task:** #100585 (Phase 0' ADR Consolidation — EInvoiceAdapter lifecycle)
**Supersedes:** ADR-016 v1 (2026-05-11, MC #100362) — this is the authoritative version
**Cross-references:**

- ADR-015 (CountryPlugin — `generateEInvoiceXml()` and `submitToFiscalPlatform()` delegate
to adapters)
- ADR-019 (Integration Adapter Registry — `AdapterConfig`, secret taxonomy, categories)
- ADR-023 §3.3 (backend country differentiation — market selected before adapter dispatch)
- `apps/api/src/main/kotlin/no/alai/bilko/einvoice/EInvoiceTypes.kt` (canonical types on
disk)
- `apps/api/src/main/kotlin/no/alai/bilko/country/hr/StorecoveHrFiskEInvoiceAdapter.kt` (HR
reference)
- Plan v3 §4b ADR-016 requirement + §4d HR critical path — `~/system/specs/bilko-multi-
market-architecture-plan-v3-2026-05-11.md`

---

## 1. Context

### 1.1 The Four-Platform Problem

Bilko targets four tax jurisdictions with four incompatible e-invoice fiscal platforms:

| Market | Platform                                | Transport  | Format                        |
Status          |
| ------ | --------------------------------------- | ---------- | ----------------------------- |
--------------- |
| HR     | HR-FISK / FINA via Storecove            | Peppol AS4 | UBL 2.1 + HR CIUS             |
STUB (MC #8675) |
| RS     | SEF (efaktura.gov.rs)                   | REST API   | SEF XML (Serbian-specific)    |
Phase 1S        |
| BA-FED | CPF (Centralna platforma za fakture)    | TBD ~2027  | TBD                           |
Phase 1B        |
| BA-RS  | UINO (stub name)                        | TBD        | TBD                           |
Phase 1B        |

Without a canonical abstraction, each platform's integration detail bleeds into the core
invoice service — reproducing the Variant B coupling problem (ADR-bilko-002 §3).

### 1.2 Existing Types on Disk (verified 2026-05-11)

`EInvoiceTypes.kt` already defines (lines 1–224):

- `AdapterLifecycleState` enum: `STUB`, `SANDBOX_VERIFIED`, `PRODUCTION`
- `EInvoiceStatus` enum: `PENDING`, `APPROVED`, `REJECTED`, `CANCELLED`, `ERROR`
- `InvoiceTypeCode`: UNTDID codes 380, 381, 383, 384
- `Address`, `PartyInfo` / `Party` typealias
- `PaymentMeans`: `paymentMeansCode`, `paymentReference`, `iban`
- `TaxCategory` enum: `S, Z, E, K, G, O, AE` per EN 16931 BT-118
- `TaxBreakdown`, `InvoiceLine`, `CanonicalInvoice`, `SubmitResult`, `InvoiceTotals`
- `EInvoiceAdapter` interface with 4 methods + 2 properties

`AdapterTypes.kt` (in `no.alai.bilko.adapter`) defines:

- `AdapterErrorCode` enum with 10 codes including `NOT_IMPLEMENTED`
- `AdapterException(code, market, retryable, rawPayload, message, cause)`
```

The `EInvoiceAdapter` interface and lifecycle states exist but are not formally documented.
`StorecoveHrFiskEInvoiceAdapter` implements the interface — `serialize()` is fully operational
offline; all other methods throw `NOT_IMPLEMENTED`. This ADR formalises the contract and lifecycle governance.

---

2. Decision

2.1 EInvoiceAdapter Interface — Formal Contract

Defined in `apps/api/src/main/kotlin/no/alai/bilko/einvoice/EInvoiceTypes.kt` lines 200–224.
Reproduced here as the normative specification with full contract annotations:

```kotlin
interface EInvoiceAdapter {
    val jurisdiction: TaxJurisdiction
    val lifecycleState: AdapterLifecycleState

    /**
     * Serialize a canonical invoice to the adapter-specific wire format.
     *
     * CONTRACT:
     * - MUST be offline-capable — no network, no credentials required.
     * - MUST be deterministic: same [invoice] input produces identical bytes.
     * - MUST NOT log raw PII fields (OIB, IBAN, document_data) — call sanitizeForLog().
     * - Returns the full wire-format payload for the platform:
     *     HR: Storecove JSON envelope wrapping UBL 2.1 XML
     *     RS: SEF XML (Serbian Ministry of Finance schema)
     *     BA: CPF/UINO platform format (TBD)
     * - Throws AdapterException(VALIDATION_BUSINESS_RULE) for constraint violations
     *   (non-EUR currency for HR, invalid OIB, empty lines, etc.)
     * - AdapterConfig.enabled check is NOT performed here — callers check before invoking.
     * - This method is ALWAYS available, even in STUB lifecycle.
     */
    fun serialize(invoice: CanonicalInvoice): ByteArray

    /**
     * Submit the serialized invoice bytes to the fiscal platform.
     *
     * CONTRACT:
     * - Requires live credentials (API key, OAuth token, or certificate).
     * - MUST include an idempotency key (platform-specific — see §2.3).
     * - Returns SubmitResult on success; throws AdapterException on ALL failures.
     * - NEVER propagates platform-native exceptions (Ktor ResponseException, etc.) —
     *   map every platform exception to AdapterException before propagating.
     * - Implementations in STUB lifecycle MUST throw NOT_IMPLEMENTED (see §2.5).
     * - Idempotency: platforms may return 409 DUPLICATE on re-submission.
     *   Caller should treat 409 as success — extract submission ID from error body.
     *
     * @param serializedInvoice bytes from serialize()
     * @param invoice original CanonicalInvoice (needed for idempotency key generation)
     */
    fun submit(serializedInvoice: ByteArray, invoice: CanonicalInvoice): SubmitResult

    /**
     * Poll the fiscal platform for the current status of a submitted invoice.
     *
     * CONTRACT:
     * - [submissionId] is SubmitResult.platformInvoiceId from submit().
     * - Returns current EInvoiceStatus.
     * - This method is IDEMPOTENT — safe to call multiple times with the same submissionId.
     * - Callers implement exponential backoff; this method does NOT retry internally.
     * - Implementations in STUB lifecycle MUST throw NOT_IMPLEMENTED (see §2.5).
     * - NEVER log rawPayload without sanitizeForLog().
     */
    fun pollStatus(submissionId: String, invoice: CanonicalInvoice): EInvoiceStatus

    /**
     * Parse an inbound invoice from a raw fiscal platform webhook payload.
     *
```

```
     * CONTRACT:
     * - [rawPayload] is the raw bytes from the platform webhook (Storecove POST, SEF
callback).
     * - Returns CanonicalInvoice with adapterMetadata populated for platform-specific
fields:
     *     HR: "hr.supplierOib", "hr.buyerOib", "hr.pozivNaBroj"
     *     RS: "rs.supplierPib", "rs.buyerPib", "rs.sefId"
     * - Implementations in STUB lifecycle MUST throw NOT_IMPLEMENTED (see §2.5).
     * - NEVER log rawPayload before passing through sanitizeForLog().
     * - parseIncoming() is deferred for HR: not required for v1 HR GA (Phase 1H.6 scope).
     *   Implement 90 days post-GA (see Plan v3 §4d).
     */
    fun parseIncoming(rawPayload: ByteArray): CanonicalInvoice
}
```

2.2 CanonicalInvoice — EN 16931 Subset

The internal invoice representation, independent of any platform wire format.
Defined in `EInvoiceTypes.kt` lines 141–156:

```kotlin
data class CanonicalInvoice(
    val id: String,                          // Internal UUID — Storecove document_id (D2
dedup)
    val invoiceNumber: String,               // BT-1: human-readable invoice number
    val issueDate: LocalDate,                // BT-2
    val dueDate: LocalDate,                  // BT-9
    val typeCode: InvoiceTypeCode,           // BT-3: UNTDID 1001 (380/381/383/384)
    val currencyCode: String,                // BT-5: ISO 4217 ("EUR", "RSD", "BAM")
    val jurisdiction: TaxJurisdiction,       // Routing discriminator (non-EN16931)
    val supplier: PartyInfo,                 // BG-4: name, taxId (OIB/PIB/JIB), address
    val buyer: PartyInfo,                    // BG-7: same structure
    val lines: List<InvoiceLine>,            // BG-25: quantity, unitPrice, lineTotal,
taxRate
    val taxBreakdowns: List<TaxBreakdown>,   // BG-23: one entry per rate band
    val paymentMeans: PaymentMeans? = null,  // BG-16: paymentMeansCode, IBAN, reference
    val note: String? = null,                // BT-22: free text note
    val adapterMetadata: Map<String, String> = emptyMap(), // platform-specific extras
)
```

Field constraints:

Field	Constraint	Enforced in
`currencyCode`	"EUR" for HR (HALT-3 — Croatia adopted EUR 2023-01-01)	serialize() HR
`supplier.taxId`	OIB (HR, 11-digit ISO 7064 MOD 11,10) / PIB (RS, 9-digit) / JIB (BA, 13-digit)	serialize() per market
`lines`	Non-empty — EN 16931 §BG-25 minimum one line	serialize()
`taxBreakdowns`	Must sum to lines.(taxRate * lineTotal) — tolerance 0.01	InvoiceService
`adapterMetadata`	HR inbound: `hr.supplierOib`, `hr.buyerOib`, `hr.pozivNaBroj`	parseIncoming()

What CanonicalInvoice is NOT:

- Not a DB entity (mapped from `invoices` + `invoice_items` tables on read)
- Not a REST API DTO (API layer maps separately)
- Not versioned independently — evolves with EN 16931 minor revisions

2.3 Adapter Lifecycle State Machine

Defined in `EInvoiceTypes.kt` lines 22–26. Transition criteria formalised here:

```
STUB
  ? Compiles. All 3 network methods throw NOT_IMPLEMENTED.
  ? serialize() MAY be operational (HR: already works offline).
```

```
  ? AdapterConfig row not required.
  ?
  ? Transition criteria ? SANDBOX_VERIFIED:
  ?   1. Provider account provisioned (MC #8675 for HR/Storecove)
  ?   2. Credentials loaded in GCP Secret Manager (see §2.6 secret taxonomy)
  ?   3. 5 sandbox test invoice types pass with REAL platform submission IDs (§2.4)
  ?   4. pollStatus() confirmed for each submitted invoice
  ?   5. Proveo evidence file with submission IDs uploaded to BookStack
  ?   6. lifecycleState field updated to SANDBOX_VERIFIED in adapter source
  ?
  ?
SANDBOX_VERIFIED
  ? All 4 methods operational against provider sandbox.
  ? AdapterConfig(market, EINVOICE, enabled=true) in STAGE DB.
  ?
  ? Transition criteria ? PRODUCTION:
  ?   1. Securion audit: adapter error handling + PII sanitization (see §2.7)
  ?   2. 30 continuous days on STAGE Cloud Run with zero
  ?      AdapterErrorCode.PLATFORM_INTERNAL_ERROR alerts
  ?      (Prometheus metric: bilko_integration_request_total)
  ?   3. AdapterConfig(market, EINVOICE, enabled=true) in PRODUCTION DB
  ?   4. CEO sign-off (this is the go-live gate)
  ?
  ?
PRODUCTION
  ? Live. All 4 methods operational against production platform.
  ? Incident response: if critical error rate > 5% over 15min window,
  ?   automated alert ? Slack #bilko-incidents ? human decision to flip
  ?   AdapterConfig.enabled = false (no redeploy needed).
```

Current HR state (2026-05-13): STUB

- `serialize()`: WORKS (offline). Unit-tested.
- `submit()`: throws NOT_IMPLEMENTED — MC #8675 pending
- `pollStatus()`: throws NOT_IMPLEMENTED
- `parseIncoming()`: throws NOT_IMPLEMENTED (deferred post-GA)

2.4 HR-FISK Storecove Sandbox Validation Matrix

5 invoice types required for SANDBOX_VERIFIED transition. All must produce real Storecove submission GUIDs (not mock strings). Proveo (Angie Jones) runs these tests.

#	Invoice Type	UNTDID Code	Scenario	Expected Storecove Response	Evidence Required
1	B2B outbound commercial	380	Supplier OIB + Buyer OIB both valid. EUR. 25% PDV. Standard commercial transaction.	HTTP 200 + `{"id": "<guid>", "status": "pending"}`	Storecove submission GUID in evidence file
2	B2G outbound	380	Buyer is HR government entity (OIB format same). `PaymentMeans.paymentMeansCode=30`.	HTTP 200 + GUID	GUID + Storecove routing.peppol.id verified as buyer OIB
3	Credit note	381	References original invoice number in `note` field. Negative line totals.	HTTP 200 + GUID	GUID + typeCode=381 confirmed in Storecove portal
4	Cancelled invoice	384	CORRECTIVE_INVOICE type. Status flow: submit ? pollStatus until APPROVED or REJECTED	HTTP 200 + GUID, then pollStatus APPROVED/REJECTED	GUID + final status confirmed
5	Inbound received	380 (inbound)	Storecove sends test webhook to Bilko's webhook endpoint. `parseIncoming()` invoked.	Webhook received. CanonicalInvoice returned. `hr.supplierOib` in adapterMetadata.	Log entry showing successful parse + extracted OIB value

HR-specific validation rules verified in each test case:

- `currencyCode = "EUR"` (HALT-3)

- Supplier OIB: ISO 7064 MOD 11,10 checksum valid
- Buyer OIB: ISO 7064 MOD 11,10 checksum valid
- CustomizationID: verify with Storecove support which to use (PEPPOL_BIS3 or HR_CIUS — TODO MC #8675 D3)
- `routing.peppol.scheme = "9934"` and `routing.peppol.id = <buyerOIB>`

Storecove-specific notes:

- Sandbox URL is the same as production (`api.storecove.com/api/v2`) — sandbox mode is a payload flag, not a different host. Set `STORECOVE_ENV=sandbox` env var.
- Idempotency key: SHA-256(`invoice.id` + `invoice.invoiceNumber`) ? sent as `Idempotency-Key` header.
  Platform returns HTTP 409 on duplicate — treat as success (re-fetch GUID from error body).
- `document_id` field in Storecove payload = `CanonicalInvoice.id` (Bilko UUID) — Storecove dedup key, prevents double-billing on retry (D2 in StorecoveHrFiskEInvoiceAdapter).

2.5 NOT_IMPLEMENTED Transition Rules

`AdapterErrorCode.NOT_IMPLEMENTED` is the canonical error code for STUB lifecycle methods. Rules for callers and implementers:

Implementer rules:

1. Any STUB lifecycle method that is not yet operational MUST throw:
   ```kotlin
   throw AdapterException(
       code = AdapterErrorCode.NOT_IMPLEMENTED,
       market = jurisdiction,
       retryable = false,
       rawPayload = "",
       message = "<Platform> <method> requires account — MC #<id>"
   )
   ```
2. `serialize()` is EXEMPT from the NOT_IMPLEMENTED requirement — it SHOULD be operational even in STUB lifecycle because it needs no credentials (offline contract).
3. Once an implementation moves to SANDBOX_VERIFIED, no method may throw NOT_IMPLEMENTED for the sandbox environment. If a method is genuinely deferred (e.g., `parseIncoming()` for HR v1), the lifecycle state must remain STUB until all 4 methods are operational. Exception: `parseIncoming()` for HR is formally deferred to 90 days post-GA per Plan v3 §4d. The HR adapter will hold a partial SANDBOX_VERIFIED state tracked by the `AdapterConfig` feature flag with `reason = "parseIncoming deferred — Phase 1H.6"`.

Caller rules:

1. Before calling `submit()` or `pollStatus()`, callers MUST check:
   ```kotlin
   val config = adapterConfigRepo.find(jurisdiction, "EINVOICE")
       ?: throw AdapterException(NOT_IMPLEMENTED, ...)
   if (!config.enabled) throw AdapterException(NOT_IMPLEMENTED, ..., message="Adapter disabled: ${config.reason}")
   ```
2. `NOT_IMPLEMENTED` caught at the route handler level maps to HTTP 503 (Service Unavailable)
   with body `{"error": "ADAPTER_NOT_AVAILABLE", "market": "<jurisdiction>"}`, NOT HTTP 500.
   This is the stub plugin HTTP 500 risk mitigation from ADR-015 §5.3.
3. `serialize()` callers do NOT need to check AdapterConfig — serialize is always available.

Error code precedence when multiple codes could apply:

```
NOT_IMPLEMENTED > AUTH_INVALID_CREDENTIALS > VALIDATION_BUSINESS_RULE > NETWORK_TIMEOUT
```

If a STUB adapter is also missing credentials, `NOT_IMPLEMENTED` takes precedence. Lifecycle state check happens before credential check.

2.6 Secret Management — GCP Secret Manager Taxonomy

All adapter credentials follow the taxonomy defined in ADR-019 §2.5:

```
Bilko/{env}/{market}/{secret-name}
```

```

- `{env}`: `dev`, `stage`, `prod`
- `{market}`: `HR`, `RS`, `BA_FED`, `BA_RS`
- `{secret-name}`: platform-specific identifier (kebab-case)

**HR Storecove secrets (provision after MC #8675):**

| GCP Secret Manager path | Content | Access binding |
| --- | --- | --- |
| `Bilko/stage/HR/storecove-api-key` | Storecove sandbox API key | Cloud Run SA `bilko-stage-sa` |
| `Bilko/prod/HR/storecove-api-key` | Storecove production API key | Cloud Run SA `bilko-prod-sa` |
| `Bilko/stage/HR/storecove-legal-entity-id` | Storecove legal entity ID (sandbox) | Cloud Run SA `bilko-stage-sa` |
| `Bilko/prod/HR/storecove-legal-entity-id` | Storecove legal entity ID (prod) | Cloud Run SA `bilko-prod-sa` |

**Mounting in Cloud Run:**

```yaml
# gcp-deploy.yml (Cloud Run --set-secrets pattern):
--set-secrets="STORECOVE_API_KEY=Bilko/stage/HR/storecove-api-key:latest,\
STORECOVE_LEGAL_ENTITY_ID=Bilko/stage/HR/storecove-legal-entity-id:latest"
```

Env var naming convention: `<PLATFORM>_<FIELD>`, uppercase, underscores.
Accessed in `StorecoveApiClient` via `System.getenv("STORECOVE_API_KEY")`.

Secret rotation policy:

- Rotate API keys every 90 days OR on any Storecove security notice, whichever comes first.
- Previous version retained in Secret Manager for 24h to allow graceful failover.
- Rotation event: create new secret version ? update Cloud Run env ? verify health endpoint ? delete previous version 24h later.

Never in source code or logs: API keys, legal entity IDs, OIB values, IBAN values. `StorecoveHrFiskEInvoiceAdapter.sanitizeForLog()` must be called on all Storecove response bodies before logging.

RS future secrets (Phase 1S):

GCP Secret Manager path	Content
`Bilko/stage/RS/sef-api-key`	SEF sandbox access token
`Bilko/prod/RS/sef-api-key`	SEF production access token
`Bilko/stage/RS/sef-username`	SEF API username
`Bilko/prod/RS/sef-username`	SEF API username (prod)

SEF uses OAuth2 with client credentials. The token endpoint is `https://efaktura.mfin.gov.rs/`
(Serbian Ministry of Finance). Exact credentials shape to be confirmed at Phase 1S kickoff.

2.7 Per-Platform Field Mapping

How `CanonicalInvoice` fields map to platform-specific XML/JSON:

CanonicalInvoice field	HR (UBL 2.1 / Peppol)	RS (SEF XML)	BA-FED	BA-RS
`supplier.taxId`	`AccountingSupplierParty/.../CompanyID @schemeID="9934"` (OIB)	`/Invoice/Seller/TaxId` (PIB)	TBD	TBD
`buyer.taxId`	`AccountingCustomerParty/.../CompanyID @schemeID="9934"` (OIB)	`/Invoice/Buyer/TaxId` (PIB)	TBD	TBD
`invoiceNumber`	`cbc:ID`	`/Invoice/InvoiceNumber`	TBD	TBD
`issueDate`	`cbc:IssueDate` (ISO 8601)	`/Invoice/IssueDate`	TBD	
```

TBD |
| `typeCode.untdidCode` | `cbc:InvoiceTypeCode`
(380/381/384) | `/Invoice/InvoiceType` | TBD |
TBD |
| `currencyCode` | `cbc:DocumentCurrencyCode` + `@currencyID` on all
amounts | `/Invoice/Currency` | TBD | TBD |
| `taxBreakdowns[].taxRate` |
`TaxSubtotal/TaxCategory/Percent` |
`/Invoice/TaxTotal/TaxRate` | TBD | TBD |
| `taxBreakdowns[].taxCategory` | `TaxSubtotal/TaxCategory/ID` (S/Z/E/K per EN 16931 BT-
118) | Serbian code set | TBD | TBD |
| `paymentMeans.paymentReference` | `PaymentMeans/PaymentID` (HR "Poziv na
broj") | `/Invoice/PaymentReference` | TBD | TBD |
| `paymentMeans.iban` |
`PayeeFinancialAccount/ID` |
`/Invoice/BankAccount/IBAN` | TBD | TBD |
| `adapterMetadata` | `"hr.supplierOib"`, `"hr.buyerOib"`, `"hr.pozivNaBroj"`
(inbound) | `"rs.sefId"`, `"rs.supplierPib"` | TBD | TBD |

SEF XML note: SEF does not use UBL 2.1. It uses a Serbian-specific XML schema published
by the Ministry of Finance. The SEF adapter maps `CanonicalInvoice` ? SEF schema directly;
it does NOT go through UBL. `EInvoiceAdapter.serialize()` returns the platform's native
format.

BA adapters (Phase 1B): CPF and UINO platforms have no published API specifications as
of
2026-05-13. Phase 1B cannot begin until regulatory mandates define the technical
specification
(~2027 per plan v3 context).

2.8 HR Reference Implementation Design Decisions

`StorecoveHrFiskEInvoiceAdapter` is the reference implementation. Future adapters MUST
replicate these patterns:

| Design decision | Location in reference
impl | Rule for future adapters |
| --- |
| --------------------------------- |
| PII field redaction before logging | Lines 24–59 (`REDACT_PII_FIELDS`,
`sanitizeForLog`) | REQUIRED — GDPR / audit rules |
| Offline serialization (no credentials) | Lines 567–571
(`serialize()`) | REQUIRED per §2.1 contract |
| Idempotency key (SHA-256 of id + invoiceNumber) | Lines 591–600 (stub comment —
activate post-#8675) | REQUIRED if platform supports |
| Credential validation on startup flag | Lines 83–138 (`validateOnStartup`,
`validate()`) | REQUIRED — default false for tests |
| Error code mapping to `AdapterException` | Lines 469–515
(`StorecoveErrorMapper`) | REQUIRED — NEVER propagate native |
| Structured metrics recording (`StorecoveMetrics`) | Lines
537–540 | REQUIRED — Prometheus counters |
| Tax ID format validation in `serialize()` | Lines 748–774 (OIB
check) | REQUIRED — early error, no network |
| `document_id` for deduplication | Lines 420–437
(`StorecovePayloadBuilder.wrap()`) | REQUIRED if platform supports 409 |

---

3. Adapter Lifecycle Governance

3.1 AdapterConfig Feature Flag

All adapter network paths (`submit`, `pollStatus`, `parseIncoming`) are gated by an
`AdapterConfig` row in the database. Defined fully in ADR-019 §2.4; referenced here:

```sql
CREATE TABLE adapter_config (
    id           UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    market       VARCHAR(8) NOT NULL,   -- TaxJurisdiction enum value
    adapter_type VARCHAR(32) NOT NULL,  -- 'EINVOICE', 'BANK_STATEMENT', etc.
    enabled      BOOLEAN NOT NULL DEFAULT FALSE,
    reason       TEXT,                  -- Human-readable status note
    updated_at   TIMESTAMPTZ NOT NULL DEFAULT now(),
    UNIQUE (market, adapter_type)
);
```

```

Seed row for HR STUB state (Flyway V17):

```sql
INSERT INTO adapter_config (market, adapter_type, enabled, reason)
VALUES ('HR', 'EINVOICE', false, 'Storecove account pending — MC #8675');
```

Row is flipped to `enabled = true` by the operator (not by code) after SANDBOX_VERIFIED
transition is confirmed by Proveo evidence.

3.2 Adapter Versioning

Each adapter exposes:

```kotlin
val adapterVersion: String  // e.g. "1.0.0"
```

The `CountryPlugin` implementation declares a minimum adapter version. Incompatibility
detected at startup ? application fails fast with a clear error (not silent degradation).

---

4. Consequences

4.1 Positive

- **Offline serialization.** `serialize()` contract requires no network. Enables invoice
  PDF preview, offline testing, and regression test suites without live platform
credentials.
- **Uniform error handling.** `AdapterException` is the only exception type crossing the
  adapter boundary. Callers implement one error handler, not four platform-specific ones.
- **Lifecycle visibility.** `lifecycleState` is first-class. Dashboards show "HR adapter:
STUB"
  and alert when a market operates in degraded state.
- **Canonical model.** `CanonicalInvoice` enables cross-market reporting and analytics.
- **NOT_IMPLEMENTED ? HTTP 503.** Clients receive a clean "feature not available" response
  instead of an HTTP 500 stack trace when an adapter is in STUB state.

4.2 Negative

- **SEF XML schema maintenance.** RS's SEF format changes without semantic versioning
  guarantees. The adapter must track schema changes proactively.
- **BA adapters are TBD.** Phase 1B work cannot begin until regulations define the spec.
- **4 methods = all or nothing lifecycle.** If `parseIncoming()` is the last unfinished
  method, the adapter cannot advance to SANDBOX_VERIFIED. The HR partial-SANDBOX exception
  (§2.5 rule 3) is a pragmatic workaround; it should not become a pattern.

4.3 Risks

- **CanonicalInvoice field gap.** A platform-specific required field has no canonical
  counterpart. **Resolution:** `adapterMetadata: Map<String, String>` for platform-specific
  extras until they generalise to first-class fields.
- **Storecove CustomizationID ambiguity (D3).** Two candidate CustomizationIDs —
  PEPPOL_BIS3 and HR_CIUS. **Resolution:** Verify with Storecove support before MC #8675
  sandbox activation. This is a HALT item. Wrong choice ? all HR invoices rejected.
- **Storecove routing.network field (HALT-4).** Existing code does not include
  `routing.network` field. Verify with Storecove sandbox whether this is required.
- **Secret rotation lag.** Expired API key ? all `submit()` calls throw
  `AUTH_INVALID_CREDENTIALS`. **Mitigation:** 90-day rotation schedule + cert-expiry-
monitor
  (Task 4.3 in Plan v3).
- **OIB validation at serialize() vs submit().** Validates early (offline) but couples
  format and validation logic. Accepted trade-off: early errors are better than late ones.

---

5. References

| Reference |
Path
Lines
```

| ----- | ------- |
`EInvoiceAdapter` interface	`apps/api/src/main/kotlin/no/alai/bilko/einvoice/EInvoiceTypes.kt`	200–224
`CanonicalInvoice` definition	`apps/api/src/main/kotlin/no/alai/bilko/einvoice/EInvoiceTypes.kt`	141–156
`AdapterLifecycleState` enum	`apps/api/src/main/kotlin/no/alai/bilko/einvoice/EInvoiceTypes.kt`	22–26
`AdapterErrorCode` enum + `AdapterException`	`apps/api/src/main/kotlin/no/alai/bilko/adapter/AdapterTypes.kt`	1–41
HR reference impl — full file	`apps/api/src/main/kotlin/no/alai/bilko/country/hr/StorecoveHrFiskEInvoiceAdapter.kt`	1–777
`StorecoveMetrics` (Micrometer counters)	`apps/api/src/main/kotlin/no/alai/bilko/country/hr/StorecoveMetrics.kt`	1–73
`StorecoveApiClient` (credentials + base URL)	`StorecoveHrFiskEInvoiceAdapter.kt`	77–179
`StorecoveOibValidator` (ISO 7064 MOD 11,10)	`StorecoveHrFiskEInvoiceAdapter.kt`	194–225
`StorecoveErrorMapper` (HTTP ? AdapterErrorCode)	`StorecoveHrFiskEInvoiceAdapter.kt`	469–515
PII sanitize helper (`sanitizeForLog`)	`StorecoveHrFiskEInvoiceAdapter.kt`	24–59
`HrUblBuilder` (UBL 2.1 offline build)	`StorecoveHrFiskEInvoiceAdapter.kt`	241–387
`StorecovePayloadBuilder` (wrap JSON + dedup D2)	`StorecoveHrFiskEInvoiceAdapter.kt`	418–450
ADR-019 §2.4 (AdapterConfig table)	`docs/architecture/ADR-019-INTEGRATION-ADAPTER-REGISTRY.md`	§2.4
Plan v3 §4d HR critical path (sandbox verification)	`~/system/specs/bilko-multi-market-architecture-plan-v3-2026-05-11.md`	147–176
Plan v3 §4b ADR-016 requirement	`~/system/specs/bilko-multi-market-architecture-plan-v3-2026-05-11.md`	125–126
ADR-bilko-003 §Layer 2 (EInvoice serialization)	`~/system/specs/bilko-multi-market-architecture-plan/ADR-bilko-003-market-abstraction-layers.md`	103–117

---

6. Approval

Status: Accepted

Unblocks:

- Phase 1H Task 1H.2: `PluginHR.generateEInvoiceXml()` delegation to `StorecoveHrFiskEInvoiceAdapter`
- Phase 1H Task 1H.4: DI wiring — lifecycle state check before submit/pollStatus dispatch
- Phase 1H Task 1H.6: Storecove submit() activation (after MC #8675)
- ADR-019: Integration Adapter Registry — `AdapterConfig` table and secret taxonomy

Role	Sign	Date
Finverge — Markos Zachariadis	Signed	2026-05-13
Architecture Lead (Petter Graff)	Signed	2026-05-13
CEO (Alem Baši?)	Not required for contract ADR	—

---

7. Document History

| Date | Author | Change |

2026-05-11	Markos Zachariadis / Petter Graff — v1 — Phase 0' initial (MC #100362)
2026-05-13	Petter Graff — v2 — MC #100585: Full lifecycle state machine with explicit transition criteria; sandbox validation matrix (5 invoice types for HR-FISK Storecove); NOT_IMPLEMENTED transition rules; GCP Secret Manager taxonomy with HR+RS secret paths; HTTP 503 mapping for NOT_IMPLEMENTED; HALT items D3/D4 documented; StorecoveMetrics and StorecoveApiClient cited explicitly

ADR-017: RLS Multi-Tenancy Migration

```
# ADR-017 — RLS Multi-Tenancy Migration

**Status:** Accepted — CEO Signed 2026-05-11 (Alem Baši?). Phase 2A V17 Flyway PERMISSIVE
migration authorized for stage execution. Phase 2C RESTRICTIVE flip remains gated on
Securion audit + 30-day soak per §4 schedule.
**Date:** 2026-05-11
**Author:** Bruce Momjian (Database Architecture, CodeCraft)
**Architecture Review:** Petter Graff (CodeCraft)
**Decision-maker:** CEO Alem Baši? — SIGNED 2026-05-11 ("ok adr17 odobreno") via session
f73dafab
**Mehanik clearance:** /tmp/mehanik-cleared-100362
**MC Task:** #100362 (Phase 0' ADR Consolidation)
**Promoted from:** ADR-bilko-001 draft (`~/system/specs/bilko-multi-market-architecture-
plan/ADR-bilko-001-multi-tenant-architecture.md`)
**Cross-references:**

- ADR-023 (why single DB remains correct — §6 supersession triggers not fired; §2 context)
- ADR-015 (TaxJurisdiction enum drives `country_code` column CHECK values)
- ADR-bilko-001 (ancestor draft, fully absorbed by this ADR — do not reference ancestor)
- ADR-bilko-003 §Layer 3 (versioned CoA data model)
- Plan v3 §4a (Option D not triggered), §4c (RLS timing — PERMISSIVE before Phase 1H merge)
- `~/system/specs/bilko-multi-market-architecture-plan-v3-2026-05-11.md`

---

## 1. Context

### 1.1 Current DB State (tool-verified 2026-05-11)

| Component               |
State
                                                                  |
| ----------------------- |
---------------------------------------------------------------------------------------------
---------------------------------------------------------------- |
| Database                | `bilko-demo-db`, Cloud SQL PostgreSQL 15, europe-
north1
                 |
| Flyway migrations       | V1..V15
applied
                                                          |
| Row-Level Security      | NOT enabled — zero RLS policies on any
table
                            |
| Tenant isolation        | Application-layer only: `WHERE org_id = :principalOrgId`
clauses
          |
| `organizations.country` | Column exists; values `'RS'`, `'HR'`, `'BA'`; NOT NULL
constraint
absent
  |
| Cross-tenant leak       | Confirmed: PUT `/api/v1/invoices/{id}` and GET
`/api/v1/invoices/{id}/pdf` with cross-tenant JWT return HTTP 500 (test drift memo 2026-05-
10, Round 12.1/12.5) |

The current application-layer scoping (ADR-005) is the sole isolation mechanism. A single
missing `WHERE org_id` clause in any new route — or a refactoring that silently drops it —
is
a cross-tenant data exposure. This is not theoretical: Round 12 probes confirmed it in two
existing routes.

### 1.2 Why Single Database Remains Correct (ADR-023 §6 Check)
```

ADR-023 §6 defines the conditions that would trigger migration to Option D (per-country DBs).
All five conditions are unmet as of 2026-05-11 (Plan v3 §4a lines 100–108):

- Paying customers in 2+ markets: 0 — NOT triggered
- Regulatory request for per-country data extract: none received — NOT triggered
- HR-FISK kernel-level coupling: Storecove API path requires no kernel isolation — NOT triggered
- p95 query latency > 500ms from cross-country noise: 0 paying customers — NOT triggered
- 2 customers complain about cross-country data visibility: 0 customers — NOT triggered

Option D costs +$60/month infra and 2–4 weeks engineering per market with no customer-facing
benefit today. **This ADR is explicitly compatible with Option D migration** — RLS policies are portable to separate databases. If Option D triggers, the same policy DDL applies to each per-country DB with zero changes.

1.3 Why RLS Cannot Wait Until Post-HR GA

Plan v3 §4c (lines 135–145): the cross-tenant 500 leaks are a live security defect. With 0 paying customers today it is unexploited — but a second registered organization (required for HR demo) creates an immediately exploitable state.

RLS PERMISSIVE mode (Phase 2A) imposes zero user-facing change and zero risk of service disruption. The existing `WHERE org_id` middleware still fires, and RLS fires alongside it. Both must pass for data to be returned. A latent policy gap is caught by the application layer rather than exposing data to the wrong tenant.

CEO sign is required before Phase 2A Flyway migrations run on stage — not before this ADR document is accepted. The ADR records the decision; the sign unblocks execution.

---

2. Decision

Option C is adopted: Shared codebase, shared deployment, shared database, with PostgreSQL Row-Level Security enforcing tenant isolation.

This is the unanimous recommendation from the 5-agent architecture review (ADR-bilko-001 §framing,
line 28–30). One codebase. One Cloud Run deployment. One PostgreSQL instance with RLS.

2.1 Binding Constraints

1. `Organization.taxJurisdiction` (`TaxJurisdiction` enum `{HR, RS, BA_FED, BA_RS}` per ADR-015) is
   the primary discriminator for jurisdiction-specific behaviour.
2. `Organization.id` (UUID) is the primary tenant discriminator for data isolation.
3. RLS policies enforce data isolation at the database layer. Application code MUST NOT rely solely on `WHERE org_id = :id` clauses (ADR-005 flaw — being retired by Phase 2C).
4. The `country_code` column on `organizations` is NOT NULL with CHECK constraint `IN ('HR', 'RS', 'BA_FED', 'BA_RS')` — enforced by Flyway V16 (Phase 1H Task 1H.1).
5. EU data residency: Current `bilko-demo-db` is in Cloud SQL `europe-north1` (Finland). This IS within EU/EEA — GDPR Article 44 satisfied. Frankfurt migration (eu-central-1) is not required to unblock HR GA (Plan v3 §4d lines 179–183).

2.2 Three-Phase Migration Path

The migration is split into three phases to ensure zero service disruption and a safe rollback path at each step.

Phase 2A — PERMISSIVE RLS (parallel with Phase 1H, target: end of Week 2)

Goal: RLS policies created and attached, set to PERMISSIVE. Existing application-layer scoping continues to operate. Both layers must pass — RLS is a second check, not a replacement.

Who signs this off: CEO Alem Baši? (this ADR signature) — required before any Phase 2A Flyway migrations run on the stage database.

DDL — PERMISSIVE policies (Flyway V17):

```sql
-- V17__rls_permissive.sql
-- ZAKON: CEO sign required before this migration runs on stage.
```

```
-- Apply PERMISSIVE RLS on core tables. Application-layer WHERE org_id
-- clauses remain active. Both must pass.

-- Enable RLS on tables
ALTER TABLE organizations       ENABLE ROW LEVEL SECURITY;
ALTER TABLE invoices            ENABLE ROW LEVEL SECURITY;
ALTER TABLE invoice_items       ENABLE ROW LEVEL SECURITY;
ALTER TABLE expenses            ENABLE ROW LEVEL SECURITY;
ALTER TABLE transactions        ENABLE ROW LEVEL SECURITY;
ALTER TABLE bank_transactions   ENABLE ROW LEVEL SECURITY;
ALTER TABLE bank_accounts       ENABLE ROW LEVEL SECURITY;
ALTER TABLE accounts            ENABLE ROW LEVEL SECURITY;
ALTER TABLE contacts            ENABLE ROW LEVEL SECURITY;

-- PERMISSIVE policy: organization-scoped isolation
-- current_setting() reads the app.current_org_id session variable
-- set by the Ktor connection pool before each query (connection middleware).

CREATE POLICY org_isolation ON invoices
    AS PERMISSIVE
    FOR ALL
    TO bilko_app              -- application role (NOT superuser)
    USING (org_id = current_setting('app.current_org_id')::uuid);

CREATE POLICY org_isolation ON invoice_items
    AS PERMISSIVE
    FOR ALL
    TO bilko_app
    USING (
        invoice_id IN (
            SELECT id FROM invoices
            WHERE org_id = current_setting('app.current_org_id')::uuid
        )
    );

CREATE POLICY org_isolation ON expenses
    AS PERMISSIVE
    FOR ALL
    TO bilko_app
    USING (org_id = current_setting('app.current_org_id')::uuid);

CREATE POLICY org_isolation ON transactions
    AS PERMISSIVE
    FOR ALL
    TO bilko_app
    USING (org_id = current_setting('app.current_org_id')::uuid);

CREATE POLICY org_isolation ON bank_transactions
    AS PERMISSIVE
    FOR ALL
    TO bilko_app
    USING (
        bank_account_id IN (
            SELECT id FROM bank_accounts
            WHERE org_id = current_setting('app.current_org_id')::uuid
        )
    );

CREATE POLICY org_isolation ON bank_accounts
    AS PERMISSIVE
    FOR ALL
    TO bilko_app
    USING (org_id = current_setting('app.current_org_id')::uuid);

CREATE POLICY org_isolation ON accounts
    AS PERMISSIVE
    FOR ALL
    TO bilko_app
    USING (org_id = current_setting('app.current_org_id')::uuid);

CREATE POLICY org_isolation ON contacts
    AS PERMISSIVE
    FOR ALL
    TO bilko_app
    USING (org_id = current_setting('app.current_org_id')::uuid);
```

```
-- BYPASS for migrations and admin tooling (Flyway runs as bilko_admin)
ALTER TABLE invoices         FORCE ROW LEVEL SECURITY;
ALTER TABLE expenses         FORCE ROW LEVEL SECURITY;
ALTER TABLE transactions     FORCE ROW LEVEL SECURITY;
ALTER TABLE bank_transactions FORCE ROW LEVEL SECURITY;
ALTER TABLE bank_accounts    FORCE ROW LEVEL SECURITY;
ALTER TABLE accounts         FORCE ROW LEVEL SECURITY;
ALTER TABLE contacts         FORCE ROW LEVEL SECURITY;
-- Flyway runs as bilko_admin (superuser bypasses RLS by default).
-- Explicit FORCE is belt-and-suspenders — admin role grants BYPASSRLS if needed.

-- Set connection middleware (Kotlin Exposed / HikariCP):
-- On each connection checkout:
--   SET LOCAL app.current_org_id = '<org_uuid_from_jwt>';
-- On connection return to pool:
--   SET LOCAL app.current_org_id = '';  -- or reset_config('app.current_org_id', true)
```

Verification after Phase 2A:

```sql
-- Rogue-role test (Proveo E2E + Securion audit):
SET ROLE bilko_app;
SET LOCAL app.current_org_id = '<hr_org_uuid>';
SELECT count(*) FROM invoices;  -- must return only HR org rows
SET LOCAL app.current_org_id = '<rs_org_uuid>';
SELECT count(*) FROM invoices;  -- must return only RS org rows
-- Cross-tenant access attempt:
SET LOCAL app.current_org_id = '<hr_org_uuid>';
SELECT * FROM invoices WHERE org_id = '<rs_org_uuid>';  -- must return 0 rows (PERMISSIVE
blocks)
```

Phase 2B — Audit Log Partitioning (post-HR GA)

Goal: Partition the `logged_actions` audit table by `country_code` to enable per-jurisdiction GDPR data extraction requests and enforce per-jurisdiction retention.

```sql
-- V18__audit_log_partitioning.sql (Phase 2B — post-HR GA)

-- Declarative partitioning by country_code
CREATE TABLE logged_actions_partitioned (
    LIKE logged_actions INCLUDING ALL
) PARTITION BY LIST (country_code);

CREATE TABLE logged_actions_hr     PARTITION OF logged_actions_partitioned
    FOR VALUES IN ('HR');
CREATE TABLE logged_actions_rs     PARTITION OF logged_actions_partitioned
    FOR VALUES IN ('RS');
CREATE TABLE logged_actions_ba_fed PARTITION OF logged_actions_partitioned
    FOR VALUES IN ('BA_FED');
CREATE TABLE logged_actions_ba_rs  PARTITION OF logged_actions_partitioned
    FOR VALUES IN ('BA_RS');

-- Retention policy enforcement (aligned with CountryPlugin.getRetentionRules()):
-- HR: 11 years (Zakon o ra?unovodstvu NN 78/2015, ?l. 10)
-- RS/BA: 10 years
-- Implemented as pg_cron job deleting rows WHERE action_tstamp_tx < now() - interval '11
years'
-- per partition.

-- country_code column backfilled from organizations.country via:
-- UPDATE logged_actions SET country_code = o.country
-- FROM organizations o WHERE o.id = logged_actions.org_id;
```

RLS policy for `logged_actions` (applied in Phase 2B):

```sql
CREATE POLICY org_isolation ON logged_actions_partitioned
    AS PERMISSIVE
    FOR ALL
    TO bilko_app
```

```
    USING (org_id = current_setting('app.current_org_id')::uuid);
```

Phase 2C — RESTRICTIVE + Retire Application-Layer Scoping (post-Securion Audit)

Goal: Convert PERMISSIVE policies to RESTRICTIVE. Remove ADR-005 application-layer `WHERE org_id` middleware. RLS is the sole isolation mechanism.

Gate conditions (all must be true before Phase 2C begins):

1. Securion audit of Phase 2A policies completed — no critical findings
2. Automated rogue-role test suite passing in CI (Proveo — see Phase 2A verification above)
3. Zero cross-tenant RLS bypass incidents on stage for 30 consecutive days
4. CEO explicit sign-off for Phase 2C

```sql
-- V19__rls_restrictive.sql (Phase 2C — post Securion audit)
-- Convert PERMISSIVE ? RESTRICTIVE on all tables
-- This is the point of no return: application layer WHERE org_id is retired after this.

DROP POLICY org_isolation ON invoices;
CREATE POLICY org_isolation ON invoices
    AS RESTRICTIVE
    FOR ALL
    TO bilko_app
    USING (org_id = current_setting('app.current_org_id')::uuid)
    WITH CHECK (org_id = current_setting('app.current_org_id')::uuid);

-- Same pattern for expenses, transactions, bank_transactions, bank_accounts,
-- accounts, contacts, invoice_items (repeat for each table).
```

2.3 Versioned Chart of Accounts Table

The `chart_of_accounts` table stores jurisdiction-specific CoA entries with time-ranged validity. This supports:

- Pravilnik revisions without code changes (ADR-bilko-003 §Layer 3, lines 122–143)
- Historical invoice accuracy (rate in force at transaction date, not current rate)
- `CountryPlugin.getChartOfAccountsDefaults()` seeding on org creation (ADR-015 §2.2)

```sql
-- Part of Flyway V17 or separate V17b (Phase 2A / 1H parallel)

CREATE TABLE chart_of_accounts (
    id              UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    jurisdiction    VARCHAR(8) NOT NULL,    -- TaxJurisdiction enum value: 'HR', 'RS',
'BA_FED', 'BA_RS'
    code            VARCHAR(16) NOT NULL,   -- e.g. '1300' (HR Kontni Plan), '204' (RS
Pravilnik)
    name            VARCHAR(256) NOT NULL,
    account_type    VARCHAR(16) NOT NULL    -- ASSET, LIABILITY, EQUITY, INCOME, EXPENSE
                    CHECK (account_type IN ('ASSET', 'LIABILITY', 'EQUITY', 'INCOME',
'EXPENSE')),
    vat_treatment   VARCHAR(64),            -- e.g. 'STANDARD_RATE', 'EXEMPT', null for
non-VAT accounts
    valid_from      DATE NOT NULL,
    valid_to        DATE,                   -- NULL = currently valid
    version         INT NOT NULL DEFAULT 1, -- increments per Pravilnik revision
    notes           TEXT,                   -- statutory reference e.g. "NN 78/2015, ?l. 5"
    created_at      TIMESTAMPTZ NOT NULL DEFAULT now(),
    UNIQUE (jurisdiction, code, valid_from)
);

CREATE INDEX idx_coa_jurisdiction_date
    ON chart_of_accounts (jurisdiction, valid_from, valid_to);

-- Query pattern: entries valid on a given transaction date
-- SELECT * FROM chart_of_accounts
-- WHERE jurisdiction = $1
--   AND valid_from <= $2
--   AND (valid_to IS NULL OR valid_to > $2)
-- ORDER BY code;

-- When Croatia raises PDV from 25% to 27% on 2027-01-01:
```

```
-- INSERT INTO chart_of_accounts (jurisdiction, code, name, account_type, vat_treatment,
valid_from, version)
-- VALUES ('HR', '2400', 'PDV po stopi 27%', 'LIABILITY', 'STANDARD_RATE', '2027-01-01',
2);
-- UPDATE chart_of_accounts SET valid_to = '2026-12-31'
-- WHERE jurisdiction = 'HR' AND code = '2400' AND valid_to IS NULL AND version = 1;
-- No code change required.
```

Seeding: `CountryPlugin.getChartOfAccountsDefaults()` returns the list of entries that Flyway data migrations insert into `chart_of_accounts` for each jurisdiction. Flyway V18 (Phase 1H — separate from V17 RLS) seeds HR Kontni Plan entries.

2.4 Exchange Rate Precision Upgrade

Current precision (CLAUDE.md database rules): `NUMERIC(19,4)` for ALL monetary amounts.

Upgrade required for FX rate columns specifically:

Exchange rates require higher precision than invoice monetary amounts. Using `NUMERIC(19,4)`
for an exchange rate means EUR/RSD at 117.2350 is representable, but EUR/BAM at 1.95583 is stored as 1.9558 — a systematic rounding error that compounds across large invoice volumes and cross-currency reconciliation.

Decision: FX rate columns upgrade to `NUMERIC(20,10)`. Monetary amount columns (invoice totals, line amounts, tax amounts) remain `NUMERIC(19,4)`.

```sql
-- V17c__exchange_rate_precision.sql (Phase 2A parallel)

ALTER TABLE exchange_rates
    ALTER COLUMN rate TYPE NUMERIC(20,10);  -- was NUMERIC(19,4)

-- If an exchange_rate_history or similar snapshot table exists:
-- ALTER TABLE exchange_rate_history
--     ALTER COLUMN rate TYPE NUMERIC(20,10);

-- NEVER change invoice_items.unit_price, invoice_items.line_total,
-- transactions.amount, etc. — those remain NUMERIC(19,4).
-- Only rate/exchange_rate columns receive this upgrade.
```

Invariant: All monetary arithmetic (invoice totals, tax calculations, double-entry postings) remains at `NUMERIC(19,4)`. The precision upgrade is scoped to the FX rate storage layer only. Rounding when applying FX rates to amounts: round half-even (banker's rounding) to 4 decimal places after multiplication.

---

3. Connection Middleware — Setting `app.current_org_id`

The RLS policies use `current_setting('app.current_org_id')::uuid`. This session variable must be set on every database connection before any query executes.

Pattern (Kotlin / Exposed / HikariCP):

```kotlin
// apps/api/src/main/kotlin/no/alai/bilko/db/OrgContextInterceptor.kt (Phase 2A NEW)

/**
 * Sets the PostgreSQL session variable `app.current_org_id` to the authenticated
 * org's UUID before any database access.
 *
 * Called from the Ktor routing pipeline after JWT validation, before the
 * database transaction opens.
 *
 * Must reset after the request completes — use try/finally or Ktor plugin lifecycle.
 */
fun setOrgContext(orgId: UUID) {
    transaction {
        exec("SET LOCAL app.current_org_id = '${orgId}'")
    }
}
```

```
fun clearOrgContext() {
    transaction {
        exec("RESET app.current_org_id")
        // or: exec("SET LOCAL app.current_org_id = ''")
    }
}
```

Failure mode: If `app.current_org_id` is not set, `current_setting('app.current_org_id')` throws an error in PostgreSQL (by default). To make it return NULL instead (for Flyway admin connections that do not set the variable):

```sql
-- In V17 migration, set default:
ALTER DATABASE bilko_demo SET app.current_org_id = '';
```

And in the policy, guard against empty string:

```sql
USING (
    CASE WHEN current_setting('app.current_org_id', true) = ''
         THEN false  -- deny if not set
         ELSE org_id = current_setting('app.current_org_id', true)::uuid
    END
)
```

The `true` parameter to `current_setting()` makes it return NULL rather than throw when the variable is not set.

---

4. Migration Schedule

Phase	Flyway Version	Blocking	Target
Phase 1H.1	V16: `organizations.country` NOT NULL + CHECK	HR enum expansion (ADR-015)	ADR-015 accepted
Phase 2A	V17: PERMISSIVE RLS + CoA table + FX rate precision	Stage only	CEO sign (this ADR)
Phase 2A	V17 seed: HR Kontni Plan data	Stage only	V17 + PluginHR.getChartOfAccountsDefaults()
Phase 2B	V18: audit log partitioning	Post-HR GA	Securion review
Phase 2C	V19: RESTRICTIVE + retire ADR-005 app scoping	Post-Securion audit	Securion audit pass + CEO sign

All migrations use Flyway's expand/contract pattern. No migration modifies data in a way that cannot be reversed by a subsequent compensating migration. Backward compatibility is required across all rolling deployments.

---

5. Consequences

5.1 Positive

1. **Defence in depth.** Even if a developer introduces a missing `WHERE org_id` in a new route, RLS at the database layer prevents cross-tenant data exposure.
2. **GDPR jurisdiction extraction.** With `country_code` on `logged_actions` (Phase 2B), a request from Croatian DPA for "all data held on Croatian entities" is a single partition query, not a full-table scan with a filter.
3. **Audit surface.** Securion can review one set of RLS policies rather than auditing every application route for correct scoping.
4. **Option D readiness.** If ADR-023 §6 triggers (e.g., first paying HR customer), the same RLS DDL applies to the per-country databases without change. Migration path is not blocked by this ADR.

5.2 Negative

1. **Connection middleware requirement.** Every DB connection must set `app.current_org_id`

   before any query. Forgetting this in a new service or background job will cause all queries to return 0 rows (PERMISSIVE) or error (RESTRICTIVE). Mitigated by integration tests that verify the context middleware fires.
2. **Flyway admin bypass.** Flyway and admin tooling must run as a role that bypasses RLS (`bilko_admin` with BYPASSRLS). This role must be kept tightly restricted — it is a privilege escalation path.
3. **Phase 2A adds overhead.** Each query now evaluates an additional predicate. At current scale (0 paying customers) the overhead is immeasurable. Monitor p95 query latency after Phase 2A migration on stage.

5.3 Risks

1. **GDPR data residency.** Croatian entity data in Cloud SQL europe-north1 (Finland) is legally compliant (EU/EEA). If a future HR DPA contract specifies Frankfurt, a regional migration is required. This ADR does not block that migration.
2. **RLS policy gap.** An incorrect USING clause (e.g., JOIN condition that broadens the scope) could expose cross-tenant data. **Mitigation:** Securion audit before Phase 2C (RESTRICTIVE), automated rogue-role test in CI.
3. **Migration synchronization.** A Flyway migration failure mid-run leaves all markets degraded. All V17+ migrations must be backward-compatible and use expand/contract pattern. If V17 fails, rollback is: `DROP POLICY` + `ALTER TABLE ... DISABLE ROW LEVEL SECURITY`.

---

6. References

Reference	Path	Lines
ADR-bilko-001 (ancestor draft, absorbed by this ADR)	`~/system/specs/bilko-multi-market-architecture-plan/ADR-bilko-001-multi-tenant-architecture.md`	1–162
ADR-bilko-003 §Layer 3 (versioned CoA model)	`~/system/specs/bilko-multi-market-architecture-plan/ADR-bilko-003-market-abstraction-layers.md`	122–143
ADR-023 §6 (single-DB migration triggers — not fired)	`docs/architecture/ADR-023-TRANSITIONAL-MULTI-MARKET-ROUTING.md`	166–176
Plan v3 §4a (Option D not triggered — evidence)	`~/system/specs/bilko-multi-market-architecture-plan-v3-2026-05-11.md`	100–108
Plan v3 §4c (RLS timing — PERMISSIVE before Phase 1H)	`~/system/specs/bilko-multi-market-architecture-plan-v3-2026-05-11.md`	135–145
Plan v3 §4d (EU data residency does not block HR GA)	`~/system/specs/bilko-multi-market-architecture-plan-v3-2026-05-11.md`	179–183
ADR-015 §2.1 (TaxJurisdiction enum — `country_code` values)	`docs/architecture/ADR-015-FOUR-JURISDICTION-PLUGIN.md`	§2.1
Test drift memo (cross-tenant 500 leaks, Round 12.1/12.5)	`~/.claude/projects/-Users-makinja/memory/project_bilko_test_strategy_drift_2026-05-10.md`	—

---

7. Approval

Architecture status: Accepted (Phase 0' ADR consolidation)
CEO sign status: SIGNED 2026-05-11 — Phase 2A V17 Flyway PERMISSIVE migration authorized for stage. Phase 2C RESTRICTIVE flip remains gated on Securion audit + 30-day soak per §4 schedule.

This ADR records the architectural decision. The CEO signature below is the gate for execution of Phase 2A database migrations. It is not a gate for writing this document or for Phase 1H code work (CountryPlugin, PluginHR, DI wiring).

Role	Sign	Date
Architecture Lead (Petter Graff)	Signed	2026-05-11
Database Architecture (Bruce Momjian)	Signed	2026-05-11
CEO (Alem Baši?)	**SIGNED — session f73dafab, transcript "ok adr17 odobreno"**	2026-05-11

---

8. Document History

Date	Author	Change
2026-04-22	ALAI / ADR-bilko-001	Initial draft (multi-tenant architecture options analysis)
2026-05-11	Bruce Momjian / Petter Graff	Promoted from ADR-bilko-001 draft; ID changed to ADR-017; DDL examples added; versioned CoA DDL added; NUMERIC(20,10) FX precision noted; Phase 2B audit log partitioning added; connection middleware pattern added; CEO sign gate formalised. MC #100362.
2026-05-11	John (AI Director)	CEO Alem Baši? signed ADR-017 via session f73dafab ("ok adr17 odobreno"). Phase 2A V17 Flyway PERMISSIVE migration authorized for stage. Status header + §7 approval table updated. Unblocks Bruce Momjian dispatch for Phase 2A.

ADR-019: Integration Adapter Registry

```
# ADR-019 — Integration Adapter Registry

**Status:** Accepted
**Date:** 2026-05-11
**Author:** Petter Graff (CodeCraft — Architecture Lead)
**Decision-maker:** CEO Alem Baši?
**Mehanik clearance:** /tmp/mehanik-cleared-100362
**MC Task:** #100362 (Phase 0' ADR Consolidation)
**Cross-references:**

- ADR-015 (CountryPlugin — plugin selects adapters for its market; plugin version
compatibility)
- ADR-016 (EInvoiceAdapter — one of the 7 adapter categories; lifecycle states formalised
here)
- ADR-023 (routing — market resolved at edge before adapter dispatch)
- `apps/api/src/main/kotlin/no/alai/bilko/country/hr/StorecoveHrFiskEInvoiceAdapter.kt`
(reference impl)
- `apps/api/src/main/kotlin/no/alai/bilko/einvoice/EInvoiceTypes.kt` (AdapterLifecycleState
on disk)
- Plan v3 §6 Phase 0' Task 0'4 — `~/system/specs/bilko-multi-market-architecture-plan-v3-
2026-05-11.md`

---

## 1. Context

### 1.1 Problem: Seven Integration Surfaces, No Governance

Bilko integrates with external systems across seven functional domains. As of 2026-05-11,
only one adapter exists (`StorecoveHrFiskEInvoiceAdapter`). Without a registry and
governance
model, adding the second adapter (SEF for RS) and every subsequent adapter will produce:

1. Inconsistent error handling — platform-native exceptions leaking across boundaries
2. No feature-flag mechanism — a broken SEF adapter takes down all RS users
3. Secret sprawl — `STORECOVE_API_KEY` as an env var pattern, but no taxonomy when
   there are 7 adapters × 4 markets × 3 environments = up to 84 secrets
4. No observability standard — each adapter invents its own logging and metrics
5. No lifecycle discipline — adapters deployed to production without sandbox verification

### 1.2 Reference Implementation Patterns

`StorecoveHrFiskEInvoiceAdapter.kt` already demonstrates all the patterns this ADR
formalises. This ADR makes those patterns enforceable for all future adapters:

| Pattern                                          |
StorecoveHrFiskEInvoiceAdapter          | ADR-019 makes it |
| ------------------------------------------------ |
--------------------------------------- | ---------------- |
| PII redaction before logging                     |
| Lines 24–59 (`sanitizeForLog`)          | Mandatory        |
| `AdapterException` only (no platform exceptions) | Lines 469–516
(`StorecoveErrorMapper`)  | Mandatory        |
| Per-adapter Prometheus metrics                   |
| Lines 537–540 (`StorecoveMetrics`)      | Mandatory        |
| Lifecycle state field                            |
| Lines 547–548 (`lifecycleState =
STUB`) | Mandatory        |
| Idempotency key on submit                        |
| Lines 591–600 (D5
comment)              | Mandatory        |
| Credentials NOT required for serialize()         | Lines
567–571                           | Mandatory        |
| Startup credential validation flag               | Lines
83–138                            | Recommended      |
```

---

2. Decision

2.1 Seven Adapter Categories

Every external integration belongs to exactly one of the following categories.
Each category is a Kotlin interface in `apps/api/src/main/kotlin/no/alai/bilko/adapter/`.

Category	Interface	Purpose	Markets
1	`EInvoiceAdapter`	E-invoice serialization + fiscal platform submission	HR (Storecove), RS (SEF), BA-FED (CPF), BA-RS (UINO)
2	`CompanyRegistryAdapter`	Company data lookup (name, address, tax status) from government registries	HR (FINA), RS (APR), BA (stub)
3	`BankStatementAdapter`	Bank statement import (MT940, CAMT.053, PSD2 AISP)	All markets — via Tok Open Banking platform
4	`ExchangeRateAdapter`	FX rate feed (daily/live)	All markets (ECB primary, HNB for HR, NBS for RS)
5	`TaxFilingAdapter`	Electronic VAT/CIT return submission to tax authority	HR (ePorezna), RS (ePorezi), BA (TBD)
6	`FiscalDeviceAdapter`	Fiscal receipt device or cloud fiscal service	HR (Fiskalizacija cloud cert), RS (LPFR chip card), BA (TBD)
7	`QESSigningAdapter`	Qualified Electronic Signature for invoice signing	HR (FINA QES), RS (stub), BA (stub)

Current implementation status:

- `EInvoiceAdapter`: `StorecoveHrFiskEInvoiceAdapter` (HR, STUB lifecycle)
- All other categories: NOT YET IMPLEMENTED

2.2 Common Interface Contract

Every adapter interface extends a common `BilkoAdapter` base:

```kotlin
package no.alai.bilko.adapter

import no.alai.bilko.country.TaxJurisdiction
import no.alai.bilko.einvoice.AdapterLifecycleState

/**
 * Base contract for all Bilko integration adapters.
 *
 * Every adapter implementation MUST:
 * 1. Expose [jurisdiction] and [lifecycleState] as first-class properties.
 * 2. Throw only [AdapterException] — NEVER platform-native exceptions.
 * 3. Pass all log writes through [sanitizeForLog] (defined per-adapter for PII fields).
 * 4. Record Prometheus metrics on every external call (see §2.6).
 * 5. Not require credentials for read-only / serialization operations.
 */
interface BilkoAdapter {
    val jurisdiction: TaxJurisdiction
    val lifecycleState: AdapterLifecycleState
    val adapterVersion: String  // Semantic version string, e.g. "1.0.0"
}
```

2.3 AdapterException — Canonical Error Contract

All adapters throw `AdapterException` and nothing else. This exception type is the
single crossing point from adapter space to core service space.

```kotlin
```

```
package no.alai.bilko.adapter

import no.alai.bilko.country.TaxJurisdiction

/**
 * Canonical adapter error. The ONLY exception type that crosses the adapter boundary.
 *
 * INVARIANT: Core services catch AdapterException only. They MUST NOT catch
 * platform-native exceptions (Ktor ResponseException, HttpRequestTimeoutException,
 * java.net.SocketTimeoutException, etc.). Map those to AdapterException in the adapter.
 *
 * [retryable]: if true, caller may retry with exponential backoff.
 * [rawPayload]: sanitized (PII-redacted) raw response body for audit. NEVER raw.
 */
data class AdapterException(
    val code: AdapterErrorCode,
    val market: TaxJurisdiction,
    val retryable: Boolean,
    val rawPayload: String,
    override val message: String = code.name,
    override val cause: Throwable? = null,
) : RuntimeException(message, cause)

/**
 * Canonical error codes — adapter-independent.
 *
 * Adapters map platform-specific HTTP status codes and error bodies to these codes.
 * See StorecoveErrorMapper (lines 469–516) for the HR reference mapping.
 */
enum class AdapterErrorCode {
    // Validation errors — not retryable
    VALIDATION_SCHEMA_ERROR,       // Invalid document structure (HTTP 400/422)
    VALIDATION_BUSINESS_RULE,      // Business rule violation (e.g., invalid OIB, non-EUR
currency)
    VALIDATION_DUPLICATE_DOCUMENT, // Idempotency conflict (HTTP 409)

    // Authentication/authorisation — not retryable
    AUTH_INVALID_CREDENTIALS,      // API key invalid / token expired / certificate
rejected

    // Platform errors — retryable
    PLATFORM_RATE_LIMITED,         // HTTP 429 — back off and retry
    PLATFORM_MAINTENANCE,          // HTTP 503 — platform in scheduled maintenance
    PLATFORM_INTERNAL_ERROR,       // HTTP 5xx — transient platform error

    // Network errors — retryable
    NETWORK_TIMEOUT,               // Connection or read timeout
    NETWORK_UNREACHABLE,           // DNS resolution failure or TCP refused

    // Implementation status — not retryable
    NOT_IMPLEMENTED,               // Adapter is in STUB lifecycle state
    UNKNOWN,                       // Unmapped error; always log rawPayload for triage
}
```

Mapping rule for new adapters: Every HTTP status code the platform can return MUST have a mapping to an `AdapterErrorCode`. Use `UNKNOWN` only as a catch-all, never as the primary mapping for a known status code. See `StorecoveErrorMapper` (lines 469–516 in `StorecoveHrFiskEInvoiceAdapter.kt`) as the reference pattern.

2.4 AdapterConfig — DB-Level Feature Flag

Every adapter is gated by an `AdapterConfig` row. An adapter MUST NOT execute any network call if its `AdapterConfig.enabled = false`. This allows disabling a broken adapter without redeployment.

```sql
-- V20__adapter_config.sql (Phase 1H — during Phase 2A window)

CREATE TABLE adapter_config (
    id           UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    market       VARCHAR(8) NOT NULL,
                 -- TaxJurisdiction enum value: 'HR', 'RS', 'BA_FED', 'BA_RS'
    adapter_type VARCHAR(32) NOT NULL,
                 -- Matches the 7 categories: 'EINVOICE', 'COMPANY_REGISTRY',
```

```
                -- 'BANK_STATEMENT', 'EXCHANGE_RATE', 'TAX_FILING',
                -- 'FISCAL_DEVICE', 'QES_SIGNING'
    enabled     BOOLEAN NOT NULL DEFAULT FALSE,
    reason      TEXT,
                -- Why disabled, e.g. "MC #8675 pending — Storecove account not activated"
                -- Required when enabled=false.
    updated_at  TIMESTAMPTZ NOT NULL DEFAULT now(),
    updated_by  TEXT NOT NULL DEFAULT 'system',  -- MC task ID or admin user
    CONSTRAINT pk_adapter_config UNIQUE (market, adapter_type)
);

-- Seed: all adapters start disabled
INSERT INTO adapter_config (market, adapter_type, enabled, reason, updated_by)
VALUES
    ('HR', 'EINVOICE',         false, 'MC #8675 — Storecove account pending', 'MC-100362'),
    ('HR', 'COMPANY_REGISTRY', false, 'Not implemented — Phase 1S', 'MC-100362'),
    ('HR', 'BANK_STATEMENT',   false, 'Tok AISP integration pending', 'MC-100362'),
    ('HR', 'EXCHANGE_RATE',    false, 'ECB feed not configured', 'MC-100362'),
    ('HR', 'TAX_FILING',       false, 'ePorezna integration Phase 2 scope', 'MC-100362'),
    ('HR', 'FISCAL_DEVICE',    false, 'Fiskalizacija cert not configured', 'MC-100362'),
    ('HR', 'QES_SIGNING',      false, 'FINA QES Phase 2 scope', 'MC-100362'),
    ('RS', 'EINVOICE',         false, 'SEF adapter Phase 1S scope', 'MC-100362'),
    ('BA_FED', 'EINVOICE',     false, 'CPF platform TBD ~2027', 'MC-100362'),
    ('BA_RS', 'EINVOICE',      false, 'UINO platform TBD', 'MC-100362');
-- (Remaining BA/RS adapter rows follow same pattern)

-- Admin can enable without redeploy:
-- UPDATE adapter_config SET enabled = true, reason = NULL, updated_by = 'MC-8675-DONE'
-- WHERE market = 'HR' AND adapter_type = 'EINVOICE';
```

Kotlin enforcement pattern:

```kotlin
// In the adapter registry
(apps/api/src/main/kotlin/no/alai/bilko/adapter/AdapterRegistry.kt)
fun requireEnabled(market: TaxJurisdiction, adapterType: String) {
    val config = adapterConfigRepository.find(market, adapterType)
        ?: throw AdapterException(
            code = AdapterErrorCode.NOT_IMPLEMENTED,
            market = market,
            retryable = false,
            rawPayload = "",
            message = "No AdapterConfig row for ($market, $adapterType) — run Flyway V20"
        )
    if (!config.enabled) {
        throw AdapterException(
            code = AdapterErrorCode.NOT_IMPLEMENTED,
            market = market,
            retryable = false,
            rawPayload = "",
            message = "Adapter ($market, $adapterType) is disabled: ${config.reason}"
        )
    }
}
```

2.5 Lifecycle States and Transition Criteria

Formalised from `AdapterLifecycleState` enum in `EInvoiceTypes.kt` lines 22–26,
and from ADR-016 §2.3. Applies to ALL adapter categories.

```
STUB ????????????????????? SANDBOX_VERIFIED ????????????????????? PRODUCTION
```

STUB (initial state for all adapters):

- Compiles and registers successfully
- All network methods throw `AdapterException(code=NOT_IMPLEMENTED)`
- `serialize()` / read-only operations may work (e.g., HR serialize works in STUB)
- `AdapterConfig.enabled` is `false`

SANDBOX_VERIFIED transition criteria (all must be true):

- Minimum 5 distinct happy-path test cases pass against the real sandbox platform
  (not mocked — real submission IDs, real response payloads)
- All test case submission IDs are archived in BookStack as evidence
- Error mapping verified: at least HTTP 400, 401, 409, 429, 503, 5xx all produce
  correct `AdapterErrorCode` values (not `UNKNOWN`)
- Proveo sign-off with evidence file path in MC task
- `AdapterConfig.enabled` can be set to `true` after this point

PRODUCTION transition criteria (all must be true):

- `SANDBOX_VERIFIED` already achieved
- Securion review of adapter error handling, PII sanitization, and idempotency key
  implementation — no critical findings
- 30 consecutive days on stage Cloud Run with:
  - Zero `PLATFORM_INTERNAL_ERROR` alerts
  - Zero cross-market routing errors
  - `bilko_integration_request_total{status="error"}` < 1% of total requests
- CEO approval for production activation
- `AdapterConfig.enabled = true` in production DB (separate row from stage DB)

2.6 Secret Taxonomy

Runtime secrets follow the pattern `Bilko/{env}/{market}/{secret-name}`.

Env first, not market first. This ensures that all production secrets are under
`Bilko/production/` and can be granted/revoked as a unit for environment promotion.

```
Bilko/
  production/
    HR/
      STORECOVE_API_KEY
      STORECOVE_LEGAL_ENTITY_ID
      FINA_QES_CERTIFICATE          (Phase 2)
      EPOREZNA_CLIENT_SECRET        (Phase 2)
    RS/
      SEF_API_KEY                   (Phase 1S)
      LPFR_DEVICE_CERT              (Phase 2)
    BA_FED/
      CPF_API_KEY                   (Phase 1B — pending platform launch)
    BA_RS/
      UINO_API_KEY                  (Phase 1B — pending platform launch)
  stage/
    HR/
      STORECOVE_API_KEY
      STORECOVE_LEGAL_ENTITY_ID
    RS/
      SEF_API_KEY
    ...
  local/
    HR/
      STORECOVE_API_KEY             (developer sandbox credentials only)
    ...
```

Secret resolution hierarchy:

1. Runtime: GCP Secret Manager (current) — accessed via `SecretResolver` interface
2. Break-glass: Vaultwarden (`vault.basicconsulting.no`) — human access only, NOT runtime
source
3. Local dev: `.env.local` file (`.gitignore`'d) — NEVER committed

```kotlin
// apps/api/src/main/kotlin/no/alai/bilko/adapter/SecretResolver.kt

/**
 * Abstracts secret retrieval behind a testable interface.
 *
 * Production implementation: GCP Secret Manager.
 * Test implementation: environment variables / in-memory map.
 *
 * Secret path convention: Bilko/{env}/{market}/{secret-name}
 */
interface SecretResolver {
    /**
```

```
     * Resolves a secret value by its canonical path.
     *
     * @param path e.g. "Bilko/production/HR/STORECOVE_API_KEY"
     * @return Secret value, or null if not found.
     * @throws AdapterException(AUTH_INVALID_CREDENTIALS) if path exists but value is
empty/blank.
     */
    fun resolve(path: String): String?

    /**
     * Convenience method: builds canonical path and resolves.
     * @param env "production" | "stage" | "local"
     * @param market TaxJurisdiction enum value as string
     * @param secretName The specific secret name
     */
    fun resolve(env: String, market: String, secretName: String): String? =
        resolve("Bilko/$env/$market/$secretName")
}
```

Vaultwarden is NOT the runtime secret source. Vaultwarden is the human break-glass vault for emergency access. Do not write Kotlin code that reads from Vaultwarden at runtime. GCP Secret Manager is the runtime source.

2.7 Observability Mandate

Every adapter MUST emit the following for every network call:

Structured log line (one per call):

```
level=INFO market=HR integration=EINVOICE env=production org_id=<uuid>
  action=submit status=SUCCESS duration_ms=234 submission_id=<guid>
```

Required fields: `market`, `integration`, `env`, `org_id`. Optional but recommended: `duration_ms`, `submission_id`, `attempt` (for retries).

NEVER log:

- OIB, PIB, JIB (tax IDs)
- IBAN
- `document_data` (invoice XML body)
- `api_key`, `api_secret`

Use `sanitizeForLog()` (pattern from `StorecoveHrFiskEInvoiceAdapter.kt` lines 24–59) before any log write that touches a response body.

Prometheus metrics (one counter per adapter):

```kotlin
// apps/api/src/main/kotlin/no/alai/bilko/adapter/AdapterMetrics.kt

/**
 * Prometheus counter for all adapter network calls.
 *
 * Labels: market, integration, status (SUCCESS | ERROR | NOT_IMPLEMENTED | TIMEOUT)
 *
 * Example PromQL for HR e-invoice error rate:
 *
rate(bilko_integration_request_total{market="HR",integration="EINVOICE",status="ERROR"}[5m]
)
 *   /
 *   rate(bilko_integration_request_total{market="HR",integration="EINVOICE"}[5m])
 */
// bilko_integration_request_total{market, integration, status}
// bilko_integration_request_duration_seconds{market, integration, status}
```

Per-(market, integration) alert rule:

- Error rate > 10% over 5 minutes: PAGE (PagerDuty or Slack alert)
- Error rate > 25% over 1 minute: CRITICAL (adapter auto-disabled via `AdapterConfig`)

2.8 Adapter Versioning

Each adapter declares `val adapterVersion: String` (semantic version, e.g., `"1.0.0"`). The corresponding `CountryPlugin` implementation declares the minimum adapter version it requires:

```kotlin
// In PluginHR:
companion object {
    const val MIN_EINVOICE_ADAPTER_VERSION = "1.0.0"
}

// Startup check in DI.kt:
val adapter = StorecoveHrFiskEInvoiceAdapter()
require(semVer(adapter.adapterVersion) >= semVer(PluginHR.MIN_EINVOICE_ADAPTER_VERSION)) {
    "PluginHR requires EInvoiceAdapter >= ${PluginHR.MIN_EINVOICE_ADAPTER_VERSION}, got ${adapter.adapterVersion}"
}
```

Adapters are versioned independently of the `CountryPlugin`. Breaking changes to an adapter interface (e.g., new required parameter in `submit()`) require a major version bump and a coordinated plugin + adapter update.

2.9 Idempotency Requirements

All submit-type methods in all adapters MUST include an idempotency key.

The idempotency key format is adapter-specific, but the value MUST be derived deterministically from the invoice or entity content — never a random UUID.

Adapter	Method	Idempotency key derivation
EInvoiceAdapter (HR)	`submit()`	`SHA-256(invoice.id + invoice.invoiceNumber)` — matches Storecove D5
EInvoiceAdapter (RS)	`submit()`	`SHA-256(invoice.id + invoice.invoiceNumber)` (same pattern)
TaxFilingAdapter	`submit()`	`SHA-256(filing.periodStart + filing.periodEnd + org.taxId)`
QESSigningAdapter	`sign()`	`SHA-256(document.contentHash + signer.taxId)`

Rationale: A network timeout after the platform receives the request but before the response arrives will cause the client to retry. Without idempotency, this creates a duplicate document. Storecove returns HTTP 409 on duplicate `document_id` (D2 in `StorecovePayloadBuilder.wrap()` lines 420–436) — the pattern must be replicated.

---

3. Implementation Path

Phase	Task	Deliverable	Status
Phase 0'	This ADR written to disk	`ADR-019-INTEGRATION-ADAPTER-REGISTRY.md`	DONE
Phase 1H	`AdapterException` + `AdapterErrorCode` formalized	`adapter/AdapterException.kt` (already exists — verify package)	Verify existing
Phase 1H	`AdapterConfig` Flyway migration (V20)	`V20__adapter_config.sql`	BLOCKED BY Phase 2A
Phase 1H	`SecretResolver` interface + GCP impl	`adapter/SecretResolver.kt` + `GcpSecretResolver.kt`	Phase 1H.4+
Phase 1H	`AdapterRegistry` + `requireEnabled()` check	`adapter/AdapterRegistry.kt`	Phase 1H.4+
Phase 1H	Prometheus metrics wired in `StorecoveHrFiskEInvoiceAdapter`	`StorecoveMetrics.kt` (skeleton exists)	Phase 1H.2+
Phase 1S	SEF RS EInvoiceAdapter	`country/rs/SefRsEInvoiceAdapter.kt`	Post-HR GA
Phase 1B	CPF BA-FED + UINO BA-RS stubs	`country/ba/Cpf*`, `country/ba/Uino*`	Post-RS GA

---

4. Consequences

4.1 Positive

- **Zero platform exception leakage.** `AdapterException` as the only crossing type means core services have one error handler for all 7 × 4 = 28 potential adapter instances.
- **Hot disable without redeploy.** `AdapterConfig.enabled = false` disables a broken adapter in < 1 minute (DB write). No restart required. Incident response time drops from minutes (redeploy) to seconds (DB update).
- **Observability from day one.** Every adapter emits standardized metrics. Error rate alerts fire before users report issues.
- **Secret hygiene.** Env-first taxonomy (`Bilko/{env}/{market}/{secret}`) makes environment promotion (stage ? production) a structured operation, not an ad-hoc copy. Break-glass access is separated from runtime access.

4.2 Negative

- **Adapter scaffolding cost.** Each new adapter requires implementing the full interface contract, `AdapterConfig` rows, `SecretResolver` wiring, and Prometheus metrics. Estimate: 1–2 days for a new adapter in STUB state.
- **AdapterConfig is a deployment dependency.** The application fails at startup if `adapter_config` rows do not exist. Flyway V20 must run before the application version that adds `requireEnabled()` checks.

4.3 Risks

- **`AdapterConfig` DB unavailable.** If the database is unreachable, `requireEnabled()` fails, blocking all adapters. Mitigation: cache `AdapterConfig` in-memory at startup with a TTL of 5 minutes. Use cached state if DB is unreachable.
- **Metrics cardinality.** High `org_id` cardinality in metrics labels would cause Prometheus memory issues. The observability mandate specifies `org_id` in log lines, NOT in Prometheus labels. Labels are `market`, `integration`, `status` only — bounded cardinality.

---

5. References

Reference	Path	Lines
`AdapterLifecycleState` enum (on disk)	`apps/api/src/main/kotlin/no/alai/bilko/einvoice/EInvoiceTypes.kt`	22–26
`StorecoveErrorMapper` (error mapping reference)	`apps/api/src/main/kotlin/no/alai/bilko/country/hr/StorecoveHrFiskEInvoiceAdapter.kt`	469–516
`StorecoveMetrics` (metrics reference)	`apps/api/src/main/kotlin/no/alai/bilko/country/hr/StorecoveHrFiskEInvoiceAdapter.kt`	537–540
PII sanitization reference	`apps/api/src/main/kotlin/no/alai/bilko/country/hr/StorecoveHrFiskEInvoiceAdapter.kt`	24–59
Idempotency key (D5) reference	`apps/api/src/main/kotlin/no/alai/bilko/country/hr/StorecoveHrFiskEInvoiceAdapter.kt`	94–98
`StorecovePayloadBuilder` (D2 document_id)	`apps/api/src/main/kotlin/no/alai/bilko/country/hr/StorecoveHrFiskEInvoiceAdapter.kt`	420–436
ADR-016 §2.3 (lifecycle states — EInvoice)	`docs/architecture/ADR-016-EINVOICE-ADAPTER.md`	§2.3
ADR-015 §2.4 (DI registration pattern)	`docs/architecture/ADR-015-FOUR-JURISDICTION-PLUGIN.md`	§2.4
Plan v3 §6 Task 0'4 acceptance criteria	`~/system/specs/bilko-multi-market-architecture-plan-v3-2026-05-11.md`	279–290

---

6. Approval

Status: Accepted
Unblocks:

```
- Phase 1H: `AdapterConfig` Flyway V20 migration
- Phase 1H: `SecretResolver` GCP implementation
- Phase 1H: Prometheus metrics wiring in `StorecoveHrFiskEInvoiceAdapter`
- Phase 1S: SEF RS adapter scaffolding (knows the contract to implement against)
- Phase 1B: CPF/UINO BA adapter stubs

| Role                              | Sign                                    | Date       |
| --------------------------------- | --------------------------------------- | ---------- |
| Architecture Lead (Petter Graff) | Signed                                  | 2026-05-11 |
| CEO (Alem Baši?)                 | Not required for registry pattern ADR | —          |

---

## 7. Document History

| Date       | Author       | Change                                                |
| ---------- | ------------ | ----------------------------------------------------- |
| 2026-05-11 | Petter Graff | Initial — Phase 0' ADR consolidation (MC #100362) |
```

ADR-020: Backend Canonical — Deprecate api-kotlin

```
# ADR-020: Canonical Backend is `backend/` — Deprecate `apps/api-kotlin/`

**Status:** Accepted
**Date:** 2026-04-28
**Author:** ALAI, 2026
**Related:** ADR-009 (superseded), ADR-011, ADR-015, ADR-016, ADR-017, ADR-018, ADR-019

> **MAJOR PATH UPDATE (2026-04-29):** `backend/` ? `apps/api/` (canonical Kotlin/Ktor
location now). `apps/api-legacy/` ? `.archive/api-legacy/`. The deprecation of `api-
kotlin/` in this ADR was executed in MC #10034 (deleted as `apps/api-kotlin-abandoned`).
See ADR-021.

---

## Context

### The Dual-Backend Incident

As of 2026-04-27, the Bilko repository contained two parallel, independent Kotlin/Ktor
backends
serving identical purposes — an architectural anomaly discovered during forensic audit MC
#9892.

**Timeline — git-verified (SHA + date + author):**

| SHA       | Date       | Author  |
Event
                                                                                  |
| --------- | ---------- | ------- |
-------------------------------------------------------------------------------------
---------------------------------------------------------------------- |
| `5f97eff` | 2026-03-04 | John AI | Earliest backup commit — `backend/` fully present with
`build.gradle.kts`, package `no.alai.bilko`, Kotlin 2.3.0 / Ktor 3.4.0 / JVM
25                          |
| `e23ade3` | 2026-03-19 | Makinja | Security headers plugin and security audit added to
`backend/`
                  |
| `6b76981` | 2026-04-10 | Makinja | CI fixes applied to
`backend/`
                                                  |
| `6c71a79` | 2026-04-14 | Makinja | `apps/api-kotlin/` created — "Complete Kotlin/Ktor
backend — Auth, Invoices, Clients, Expenses, Health" — package `io.bilko`, Kotlin 2.1.20 /
Ktor 3.1.2 / JVM 21 |
| `f66ddec` | 2026-04-14 | Makinja | GCP Terraform + CI added (lands in both
directories)
                           |
| `ee27c6b` | 2026-04-15 | Makinja | `apps/api-kotlin/` scaffold finalised — 10 feature
modules, 17 source files, titled "migration
scaffold"                                                          |

**Result:** `backend/` was first on 2026-03-04 (6 weeks before `apps/api-kotlin/`). A
generic
builder agent (dispatched without Mehanik gate clearance) created `apps/api-kotlin/` on
2026-04-14 while `backend/` was already the active implementation. From 2026-04-15 onward,
`apps/api-kotlin/` received no further commits. `backend/` continued active development
with
commits through 2026-04-23.

### Background — Why This Happened

CEO decision (2026-03-17, ALAI/CLAUDE.md) mandated migration of all products from
Express/TS to
Kotlin/Ktor. Migration task MC #5125 ("Bilko backend: Express/TS ? Kotlin/Ktor") was
```

created but
not formally unblocked. The Phase 1 Track A execution document
(`docs/bookstack-sync/phase1-track-a-execution.md`, lines 241 and 299) designated
`apps/api-kotlin/` as the FUTURE migration target — explicitly stating:

> "Do NOT start Track B until MC #5125 unblocks. All Track B work goes into `apps/api-kotlin/`."

However, a generic builder agent was dispatched into `apps/api-kotlin/` on 2026-04-14 before
MC #5125 was formally unblocked and before Mehanik clearance was obtained. This created an unauthorized parallel scaffold while the actual canonical domain implementation continued to grow
in `backend/`.

State at Time of Discovery (MC #9892, 2026-04-27)

`backend/`:

- Package: `no.alai.bilko`
- Kotlin 2.3.0 / Ktor 3.4.0 / JVM 25
- 51 `.kt` source files + 3 test files
- Full domain: HR-FISK, SEF, EInvoice, AdapterException (14 codes), Koin DI, Redis, Apache PDFBox, Sentry, EmailService, SecurityHeaders, CORS, RateLimit
- All ADR-015 through ADR-019 reference `no.alai.bilko` paths inside `backend/`
- Last commit: 2026-04-23 (John, active)

`apps/api-kotlin/`:

- Package: `io.bilko`
- Kotlin 2.1.20 / Ktor 3.1.2 / JVM 21
- 17 `.kt` source files + 3 test files
- Skeleton only: Auth, DB tables, feature route scaffolds
- Missing: HR-FISK, SEF, EInvoice adapter interface, AdapterException, Koin DI, Redis, PDFBox, Sentry, EmailService, RateLimit, CORS
- Last commit: 2026-04-15 `ee27c6b` (Makinja, **13 days stale** at discovery)
- NOT deployed (confirmed by `docs/evidence/9386/verification.json` line 36 and `docs/evidence/9398/verify-cookie-fix.js` line 76)
- NOT referenced in any architecture document

Prior Audit Gap

The preliminary architecture audit (2026-04-27) saw `apps/api-kotlin/` in the directory listing
and noted: _"need to confirm these are indeed empty/removed"_ — but did not follow through with a
`find` verification. The tool-first discipline (ZAKON NULA) required explicit verification before
any assumption about directory contents. The audit concluded without detecting the 17 active
Kotlin files, full auth module, and complete Gradle build in `apps/api-kotlin/`.

Decision

`backend/` is the canonical Kotlin/Ktor backend for Bilko.

`apps/api-kotlin/` is deprecated and will be archived as of MC #9894.

All present and future development of the Bilko API occurs in `backend/`. The `io.bilko` package namespace is abandoned. The `no.alai.bilko` namespace (established 2026-03-04) is permanent for the Kotlin backend.

Rationale

Three hard facts — all git-verified, zero assumptions:

Fact 1 — Scale disparity (51 vs 17 files).
`backend/` contains 51 Kotlin source files. `apps/api-kotlin/` contains 17. More critically,
the qualitative gap is larger than the count suggests: `backend/` contains every domain-specific

component (fiscal adapters, error registry, DI wiring, PDF generation, rate limiting, Sentry
telemetry). `apps/api-kotlin/` contains only the routing skeleton.

Fact 2 — All ADR paths reference `backend/`.
ADR-014 through ADR-019 — every architecture decision record written for this product — reference
`no.alai.bilko` paths inside `backend/`. ADR-019 was explicitly verified against
`backend/src/main/kotlin/no/alai/bilko/adapter/AdapterException.kt`. Zero architecture documents
reference `io.bilko` or `apps/api-kotlin/`. An ADR is a commitment. Reversing ADR-015 through
ADR-019 evidence chains to point at `apps/api-kotlin/` would be rework with no technical benefit.

Fact 3 — Version inversion confirms direction of travel.
`apps/api-kotlin/` is pinned to Kotlin 2.1.20 / Ktor 3.1.2 — the versions specified in the original Phase 1 Track A scaffold spec. `backend/` runs Kotlin 2.3.0 / Ktor 3.4.0 / JVM 25 —
current as of 2026-04-28. This is not a coincidence: `apps/api-kotlin/` was scaffolded once to a
fixed spec and never updated. `backend/` was actively maintained and upgraded. The version delta
tells the entire story: one codebase is alive, the other is frozen at its creation point.

Consequences

For Lane B BLOCKER Tasks (#9852 / #9853 / #9854 / #9855)

All Lane B backend tasks are unblocked against `backend/` as the target. No work should be directed to `apps/api-kotlin/`. Any task whose scope referenced `apps/api-kotlin/` or `io.bilko`
must be updated to reference `backend/` and `no.alai.bilko` before execution begins.

For MC #5125 (Bilko Express ? Kotlin Migration)

MC #5125 was the formal trigger for the Kotlin migration and the stated prerequisite for any
work in `apps/api-kotlin/`. With this ADR:

- `backend/` fulfills the Kotlin/Ktor migration requirement — the migration is structurally complete at the backend layer.
- MC #5125 may be closed (DONE) once BUILD-BLUEPRINT.md is updated (MC #9897) to document `backend/` as the canonical backend and the Express legacy (`apps/api-legacy/`) as deprecated.
- The specific Track A intent ("apps/api-kotlin/ is the migration landing zone") is superseded by this ADR. Track B work proceeds in `backend/` directly.

For BUILD-BLUEPRINT.md

BUILD-BLUEPRINT.md §3 currently documents `apps/api/` (now `apps/api-legacy/`) as the backend
and makes no mention of either `backend/` or `apps/api-kotlin/`. This is pre-migration documentation. MC #9897 (BUILD-BLUEPRINT update) must:

1. Replace the backend section to reference `backend/` (package `no.alai.bilko`, Kotlin 2.3.0,
   Ktor 3.4.0)
2. Document the directory structure: `backend/` lives outside Turborepo workspace (independent
   Gradle + GCP Cloud Run deploy)
3. Update build commands to `cd backend && ./gradlew run`
4. Mark `apps/api-legacy/` as deprecated with a pointer to its decommission timeline

For DEPLOY-MAP.md

DEPLOY-MAP.md currently records bilko-api as "Manual only (Kotlin TBD)". After Dockerfile work
(MC #9898), this entry must be updated to reflect: source = `backend/`, build = Docker multi-stage, deploy target = GCP Cloud Run via `gcp-deploy.yml`.

Negative Consequences

1. **One-time porting effort.** `apps/api-kotlin/` contains a more rigorous implementation of
   refresh token rotation (`features/auth/AuthRepository.rotateRefreshToken()`). This pattern
   should be reviewed against `backend/` before archiving — MC #9895 covers this comparison.

2. **Track A plan invalidated.** The Phase 1 Track A execution document explicitly designated
   `apps/api-kotlin/` as the future target. That plan is now superseded. The document must be
   annotated with a pointer to this ADR to prevent future agents from acting on stale guidance.

3. **Feature-sliced architecture not adopted.** `apps/api-kotlin/` used a feature-sliced layout
   (`features/auth/`, `features/invoices/`). `backend/` uses a layered layout (`routes/`,
   `services/`, `auth/`). The architectural pattern debate is resolved in favor of the layered
   approach by inertia — 51 files are not being reorganized. This is a deliberate trade-off:
   stability over structural preference.

Lessons Learned

Lesson 1 — Premature Scaffold Incident

The Track A document stated that `apps/api-kotlin/` should NOT be built until MC #5125 formally
unblocked. A generic builder agent built it anyway (2026-04-14). This is the root cause of the
entire incident. The constraint in writing was not sufficient — it required a hard gate (Mehanik)
enforcing it programmatically.

Fix: Mehanik pre-dispatch gate (activated 2026-04-25, MC #9274) is the structural remedy.
No backend task may be dispatched without Mehanik clearance that checks: task MC ID exists,
BUILD-BLUEPRINT.md read, scope ceiling verified, CI green if deploy.

Lesson 2 — "Probably Empty" Hallucination in Audit

The preliminary audit saw `apps/api-kotlin/` in the `ls` output and wrote "need to confirm these
are indeed empty/removed" — then concluded without verifying. The correct tool-first discipline
(ZAKON NULA) required a `find apps/api-kotlin/src -name "*.kt"` call before any assumption about
directory state. A single verification command would have revealed 17 Kotlin files and flagged
the duplicate immediately.

Fix: Any directory flagged as "need to confirm" in an audit is an open obligation, not a
closed finding. Audits are not complete until all flagged items are machine-verified.
Post-audit review by a second agent (MC #9892 forensic) should be standard for architecture-level
audits on active codebases.

Lesson 3 — ZAKON NULA Violation (Tool-First)

John dispatched the backend-dev agent to `apps/api-kotlin/` without reading BUILD-BLUEPRINT.md,
without running `node ~/system/tools/mc.js show 5125`, and without querying the existing project
structure. Had BUILD-BLUEPRINT.md been read first, it would have been apparent that `backend/`
was the active implementation and that `apps/api-kotlin/` was the designated (but not yet active)
future target — a distinction requiring a human (Alem) decision, not an agent initiative.

Fix: ZAKON NULA (CLAUDE.md) is enforced by the Mehanik pre-dispatch hook. The hook

requires
tool-verified project state before clearing any build dispatch.

Lesson 4 — ZAKON #1 Violation (Specialist Routing)

MC #5125 is a complex backend migration (Express/TS ? Kotlin/Ktor, domain logic, multi-market
fiscal adapters, DI framework selection). This requires a **specialist** — CodeCraft (Petter Graff
/ Hadi Hariri), not a generic builder agent. CLAUDE.md §5 is unambiguous: "Never generic builder/minion. Route to the right company." Generic agents lack the architectural judgment to
navigate this class of decision (where does the backend live? which package namespace? which
version pins?).

Fix: Complex backend migrations are categorically CodeCraft work. If the specialist routing
table in CLAUDE.md is unclear for a given task, the correct action is to ask John, not to default
to a generic pool. The Mehanik gate now enforces specialist routing as part of its clearance
criteria.

Migration Path

The following tasks (C2–C6, MC #9894–#9898) execute the deprecation and consolidation. All tasks
have Mehanik-cleared MC IDs. Sequencing matters: C2 (archive) must complete before C3 (port auth) to avoid confusion about which directory to edit.

C2 — Archive `apps/api-kotlin/` (MC #9894)

Owner: CodeCraft | **Effort:** S (1h)

1. Rename `apps/api-kotlin/` to `apps/api-kotlin-abandoned/`.
2. Add `README.md` at the root of the renamed directory:
   ```
   DEPRECATED 2026-04-28 — see ADR-020
   This directory is the abandoned migration scaffold created 2026-04-14 to 2026-04-15.
   The canonical Kotlin backend is /backend/ (no.alai.bilko, Kotlin 2.3.0, Ktor 3.4.0).
   Do not edit this directory. It will be deleted after 2026-05-28.
   ```
3. Verify `turbo.json` and root `package.json` workspaces do NOT include `apps/api-kotlin` or
   `apps/api-kotlin-abandoned` (Turborepo workspace scope must not resolve against it).
4. Annotate `docs/bookstack-sync/phase1-track-a-execution.md` lines 241 and 299 with:
   `[SUPERSEDED by ADR-020 — apps/api-kotlin abandoned, backend/ is canonical]`.

C3 — Port Auth Improvements to `backend/` (MC #9895)

Owner: CodeCraft | **Effort:** M (4h)

Compare `apps/api-kotlin-abandoned/features/auth/AuthRepository.kt` (specifically `rotateRefreshToken()` and the ThreadLocal side-channel pattern) against `backend/src/main/kotlin/no/alai/bilko/auth/AuthService.kt`. If the abandoned version is more
rigorous, port the improvement. Do not port file structure or package names.

Scope: auth only. No feature modules, no table objects, no routing changes.

C4 — Update BUILD-BLUEPRINT.md (MC #9897)

Owner: CodeCraft | **Effort:** S (2h)

See Consequences section above for mandatory content. In addition, add an explicit architectural
note: "`backend/` lives outside the Turborepo workspace by design. It is a standalone Gradle
project with its own GCP Cloud Run deploy pipeline. Do not move it inside `apps/`."

C5 — Add Dockerfile to `backend/` + Update DEPLOY-MAP.md (MC #9898)

```
**Owner:** FlowForge | **Effort:** M (4h)

1. Port `apps/api-kotlin-abandoned/Dockerfile` (JVM 21, multi-stage, non-root user, health
check)
   to `backend/Dockerfile`. Upgrade base image from JVM 21 to JVM 25 (matching `backend/`
JVM
   target).
2. Verify fat JAR output name: `bilko-api.jar` (check `build.gradle.kts` shadowJar config).
3. Local build validation: `docker build -t bilko-api-test ./backend` must succeed.
4. Update DEPLOY-MAP.md bilko-api entry: source = `backend/`, Dockerfile =
`backend/Dockerfile`,
   deploy = GCP Cloud Run via `gcp-deploy.yml`.

### C6 — Proveo Verification (MC #9898 gate, Proveo)

**Owner:** Proveo (Angie Jones) | **Effort:** S (2h)

Acceptance criteria:

1. `docker build -t bilko-api ./backend` exits 0.
2. `docker run --rm -p 8080:8080 bilko-api` starts and responds to `GET /health` with HTTP
200.
3. `apps/api-kotlin/` directory no longer exists in repo root (renamed per C2).
4. `turbo.json` workspaces grep returns no match for `api-kotlin`.
5. `grep -r "io.bilko" backend/` returns no matches (no namespace contamination).

---

## References

- **MC #9892** — Forensic audit: dual Kotlin backend root-cause analysis
- **MC #9894** — C2: Archive `apps/api-kotlin/`
- **MC #9895** — C3: Port auth improvements to `backend/`
- **MC #9897** — C4: Update BUILD-BLUEPRINT.md
- **MC #9898** — C5/C6: Dockerfile + DEPLOY-MAP.md + Proveo verify
- **MC #5125** — Bilko backend migration: Express/TS ? Kotlin/Ktor (to be closed after MC
#9897)
- **ADR-015** — Four-Jurisdiction Plugin Architecture (references `no.alai.bilko`)
- **ADR-016** — E-Invoice Adapter and UBL 2.1 Canonical Model (references `no.alai.bilko`)
- **ADR-017** — RLS Multi-Tenancy (references `no.alai.bilko`)
- **ADR-018** — Market Locale Separation (references `no.alai.bilko`)
- **ADR-019** — Integration Adapter Registry (explicitly verified against `backend/` path)
- **docs/bookstack-sync/phase1-track-a-execution.md** — Phase 1 Track A intent document
(lines 241, 299 superseded by this ADR)
- **Forensic reports** — `/tmp/bilko-dual-backend-da.md`, `/tmp/bilko-dual-backend-
petter.md`

---

## Approval

**Accepted:** 2026-04-28
**Executed by:** ALAI, 2026
**Execution tasks:** MC #9894, #9895, #9897, #9898
```

ADR-021: Bilko Blueprint Section 15 Realignment

```
# ADR-021: Bilko Blueprint Section 15 Realignment

Status: Accepted
Date: 2026-04-29
Authors: ALAI (CEO Alem Basic, AI Director John)
Context: MC #10034 Phases 6-7

## Context

Bilko had drifted from ALAI Universal Blueprint Section 15. Audit revealed: 384 dirty WT
entries
(now archived), 29 unmerged branches (now archive tags), 3 sibling worktree clones, backend
at
`backend/` (blueprint expects `apps/api/`), country packages named `country-*` (blueprint
expects
`domain-*`), root contained ~33 entries (blueprint allows ~10).

CEO approved scorched-earth-lite cleanup 2026-04-28. PR #1 + PR #2 fixed CI prerequisite.
FlowForge executed Phases 0-5.

## Decision

Full Section 15 realignment in Phases 6-7:

- DELETE: `apps/api-kotlin-abandoned/`, `scratch-api/` (untracked build artifacts)
- MOVE: `backend/` ? `apps/api/` (Kotlin/Ktor canonical backend)
- RENAME: `apps/api/` (Express) ? `apps/api-express/` (Express active, migration pending)
- ARCHIVE: `apps/api-legacy/` ? `.archive/api-legacy/`
- SKIP: `landing/` ? `apps/landing/` — no git-tracked files in landing/ (only node_modules,
out)
- MOVE: `cloudbuild.yaml` ? `infrastructure/gcp/cloudbuild.yaml`
- MOVE: `docker-compose.test.yml` ? `infrastructure/docker/docker-compose.test.yml`
- MOVE: `.ci/config.yml` ? `scripts/ci/alai-ci-config.yml`
- MOVE: `ci/stubs/` ? `tools/ci-stubs/`
- MOVE: `audit-2026-04-28/` ? `docs/audits/2026-04-28/`
- MOVE: `COMPLAINT-REPORT-2026-02-20.md` ? `docs/audits/`
- MOVE: `DEPLOY-MAP.md` ? `docs/DEPLOY-MAP.md`
- MOVE: `figma-plugin/` ? `tools/figma-plugin/`
- MOVE: `design/` ? `docs/design/design/` (double-nested by git mv into existing
docs/design/)
- MOVE: `branding/` ? `docs/branding/branding/` (double-nested similarly)
- RENAME: `packages/country-{ba,ba-fed,ba-rs,hr,rs}` ? `packages/domain-{ba,ba-fed,ba-
rs,hr,rs}`

## Path Reference Updates (Phase 7)

All active code paths updated:

- `infrastructure/docker/Dockerfile.api` — updated `apps/api` ? `apps/api-express`,
`country-*` ? `domain-*`
- `infrastructure/docker/Dockerfile.api-legacy` — updated `apps/api-legacy` ?
`.archive/api-legacy`, `country-*` ? `domain-*`
- `apps/web/Dockerfile` — updated `apps/api-legacy` ? `apps/api-express`,
`packages/country-*` ? `packages/domain-*`
- `apps/api-express/package.json` — renamed to `@bilko/api-express`, deps `@bilko/country-
*` ? `@bilko/domain-*`
- `apps/api-express/vitest.config*.ts` — all path aliases updated to `domain-*`
- `apps/api-express/src/**/*.ts` — all `@bilko/country-*` imports ? `@bilko/domain-*`
- `apps/api-express/tests/**/*.ts` — all `@bilko/country-*` imports ? `@bilko/domain-*`
- `packages/domain-*/src/index.ts`, `README.md` — all self-references updated
- `packages/README.md` — updated `country-*` ? `domain-*`
- `src/shared/ubl/*.ts` — comment references updated
- `.github/workflows/gcp-deploy.yml.disabled` — path filter `backend/**` ? `apps/api/**`
- `BUILD-BLUEPRINT.md` — Section 2 (tech stack), Section 3 (project structure), Section 8
```

```
(rules)
- `CLAUDE.md` (root) — project structure and backend status sections
- `.gitleaks.toml` — updated `apps/api-legacy` ? `.archive/api-legacy`
- `package-lock.json` — deleted and regenerated clean (no stale `packages/country-*`
entries)
- `package.json` (root) — workspaces updated, lint-staged scoped to `apps/web` and
`apps/e2e` only

## Consequences

- 17 unmerged branches archived as tags; require cherry-pick MC if reactivated
- All path refs updated atomic with their respective move commits
- `package-lock.json` regenerated clean (lockfile portability ZAKON compliant)
- `gcp-deploy.yml` workflow disabled (`.disabled` suffix); re-enable deferred to Phase 8
validation
- Git history preserved via `git mv` throughout
- `lint-staged` scoped to `apps/web/**` + `apps/e2e/**` only (ESLint configs in
`packages/domain-*`
  reference a root `tsconfig.json` that does not exist; scoping prevents false failures)

## Acceptable Remaining Refs

The following historical/comment references were intentionally left as-is:

- `docs/evidence/*/verification.json` — immutable point-in-time MC evidence records
- `apps/api/src/.../CountryService.kt:11` — Kotlin comment documenting what was replaced

## Recovery

- `git checkout archive/bilko-2026-04-28-pre-cleanup` for full pre-cleanup state
- `git stash apply stash@{0}` for the 688 dirty entries (sha 8f18dc36)
- Individual branch restoration: `git checkout archive/2026-04-28/<branch>`

## Deviations from Blueprint

- `RUNBOOK.md` kept in root (mandatory per Section 15 line 1074)
- `BUILD-BLUEPRINT.md`, `CLAUDE.md`, `CHANGELOG.md`, `PIPELINE.md`, `package.json`,
`turbo.json`,
  `settings.gradle.kts`, `.env.example*` also kept in root per Section 15
- `packages/database` (current) maps to blueprint `packages/database`
- `packages/core` (Bilko-specific shared code, no direct blueprint analog) kept as-is
- `design/` and `branding/` ended up double-nested (`docs/design/design/`,
`docs/branding/branding/`)
  due to `git mv` into pre-existing `docs/design/` and `docs/branding/` subdirectories
- `apps/api-express/` is a deviation from blueprint (which expects only `apps/api/` Kotlin)
— this
  is the active Express migration target, kept alive until MC #5125 migration completes

## References

- ALAI-UNIVERSAL-BLUEPRINT.md Section 15
- MC #10027 (closed, superseded by #10034)
- MC #10034 (this work)
- MC #10044 (coverage glob fix, bundled into Phase 5d)
- ADR-020: `docs/architecture/ADR-020-BACKEND-CANONICAL-DEPRECATE-API-KOTLIN.md`
```