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

Revision #4
Created 2026-05-14 10:09:17 UTC by John
Updated 2026-06-14 20:03:15 UTC by John