Bilko ADR-015: Four-Jurisdiction Plugin

Author: ALAI, 2026

```
# ADR-015: Four-Jurisdiction Plugin Architecture

**Status:** Accepted
**Date:** 2026-04-21
**Author:** ALAI, 2026
**Replaces:** ADR-006 (single `CountryPlugin` concept, now extended to 4 jurisdictions)
**Related:** ADR-016 (E-Invoice Adapter), ADR-017 (RLS Multi-Tenancy), ADR-018 (Market vs
Locale), ADR-019 (Integration Adapter Registry)

---

## Context

Bilko must serve **four tax jurisdictions**: Serbia (RS), Croatia (HR), Bosnia-Herzegovina
Federacija (BA-FED), and Bosnia-Herzegovina Republika Srpska (BA-RS). All four share ~80% of
the accounting engine (double-entry bookkeeping, IFRS/IFRS-SME), but diverge significantly
on:

- **VAT rates**: RS 20/10%, HR 25/13/5%, BA-FED 17%, BA-RS 17%
- **Chart of accounts**: Different Pravilnik/Kontni Okvir regulations per jurisdiction
- **E-invoice platforms**: SEF (RS), HR-FISK/FINA (HR), CPF (BA-FED), UINO (BA-RS)
- **Fiscal devices**: LPFR (RS), Fiskalizacija (HR), ESET (BA-RS)
- **Filing authorities**: APR (RS), FINA (HR), UIO (BA-FED), Poreska Uprava RS (BA-RS)
- **Currencies**: RSD, EUR, BAM, BAM
- **Retention laws**: 10y (RS), 11y (HR), 10y (BA-FED), 11y (BA-RS)

### Problem Statement

How do we serve 4 jurisdictions in **one codebase**, **one deployment**, **one database**
without:

1. **Per-country forks** (Pantheon pattern — collapses team velocity)
2. **Single-jurisdiction hardcoding** (Fiken pattern — requires complete rewrite per market)
3. **Conditional spaghetti** (`if country == 'X'` throughout the core)

### Design Decision Drivers

**Expert consensus (5 agents unanimous):**

1. ONE app, ONE API, ONE DB at MVP scale — no microfrontends, no per-country deployments
2. Country plugin model (ADR-006) is architecturally sound — extend it
3. Treat BA-FED and BA-RS as **separate tax jurisdictions**, not subregions
   - Different tax authorities ? different jurisdictions
   - Different Pravilnik (chart of accounts)
   - UIO operates at entity level, not state level

**Vendor patterns adopted:**

- **Odoo `l10n_xx` module pattern** — one module per jurisdiction
- **Intuit QuickBooks "Global-by-Design"** — externalized variability config
- **Xero TaxEngine** — centralized tax engine with regional rule sets
- **SAP B1 EFM** — e-invoice adapter layer with canonical UBL 2.1

---

## Decision

### 1. Canonical Tax Jurisdiction Enum

Bosnia-Herzegovina is modeled as **two jurisdictions** because FBiH and RS entities have:

- Different tax authorities
```

- Different chart-of-accounts regulations (Pravilnik)
- Separate VAT filing authorities (UIO operates at entity level)

A single `BA` flag would require runtime branching inside the plugin — defeating the purpose of the plugin model.

```kotlin
enum class TaxJurisdiction {
    /** Serbia — SEF, RSD, PDV 20/10%, APR. */
    RS,

    /** Croatia — HR-FISK/FINA, EUR, PDV 25/13/5%, FINA. */
    HR,

    /** Bosnia-Herzegovina, Federacija entity — CPF stub, BAM, PDV 17%, UIO/FIA. */
    BA_FED,

    /** Bosnia-Herzegovina, Republika Srpska entity — UINO stub, BAM, PDV 17%, PURS. */
    BA_RS
}
```

Stored in `Organization.taxJurisdiction` (database column: `tax_jurisdiction CHAR(6)`).

2. CountryPlugin Interface Contract

All jurisdiction-specific logic **must** go through this interface. The core engine remains jurisdiction-agnostic.

Interface definition:

```kotlin
interface CountryPlugin {
    fun jurisdiction(): TaxJurisdiction
    fun calculateVat(invoice: CanonicalInvoice): VatResult
    fun generateEInvoiceXml(invoice: CanonicalInvoice): ByteArray
    fun submitToFiscalPlatform(receipt: FiscalReceipt): FiscalSubmissionHandle
    fun getChartOfAccountsDefaults(): List<ChartOfAccountEntry>
    fun getFilingDeadlines(): List<FilingDeadline>
    fun getRetentionRules(): RetentionPolicy
    fun getCurrency(): java.util.Currency
    fun getFormatters(): JurisdictionFormatters
}
```

Four implementations:

- `PluginRS` — Serbia
- `PluginHR` — Croatia
- `PluginBAFED` — Bosnia-Herzegovina Federacija
- `PluginBARS` — Bosnia-Herzegovina Republika Srpska

All implementations must be **stateless**. Plugin selection is determined by `Organization.taxJurisdiction` at runtime (read from JWT claims).

3. Package Structure

```
packages/
  country-rs/        # Serbia plugin
  country-hr/        # Croatia plugin
  country-ba-fed/    # Bosnia-Herzegovina Federacija
  country-ba-rs/     # Bosnia-Herzegovina Republika Srpska
```

Old `packages/country-ba` deprecated in Phase 1 Task 1.2.

4. Core Architectural Principles

1. **Core is jurisdiction-agnostic.** The accounting engine, invoice wizard, ledger, and reports contain **zero** country conditionals.
2. **Jurisdiction plugin owns:**
   - Tax rates and VAT calculation
   - Chart of Accounts template
   - E-invoice XML serialization (delegated to `EInvoiceAdapter`)
   - Fiscal device integration
   - Filing deadlines
   - Retention rules
3. **Canonical invoice model** (EN 16931 / UBL 2.1 subset) is internal. Each `EInvoiceAdapter` serializes to jurisdiction-specific XML.

4. **Market ? Locale.** A user in BA-RS entity uses `bs` locale; a company in Sarajevo (FBiH) also uses `bs` locale — different markets, same locale.
5. **Branding is singular.** Geographic orientation via badge + URL path prefix, not via color or typography.

Consequences

Positive

1. **Scalability:** Adding Slovenia (SI), Montenegro (ME), or North Macedonia (MK) = implement `CountryPlugin`, register in adapter registry. No core refactoring required.
2. **Testability:** Each plugin is independently testable. Jurisdiction-specific regression tests run in isolation.
3. **Compliance clarity:** Jurisdiction-specific logic is in one place — easier to audit, easier to certify.
4. **Parallel development:** Teams can work on different plugins simultaneously without merge conflicts (using worktrees).

Negative

1. **Boilerplate:** Each jurisdiction requires scaffolding 8 plugin methods + CoA data + test harness.
2. **Integration complexity:** 4 jurisdictions × 7 integration types (see ADR-019) = 28 adapters to build/maintain.
3. **Certification burden:** SEF, HR-FISK, CPF sandbox certification must pass before respective market launch (sandbox gating — Phase 1 Task 1.5).

Risks

1. **Government API stability:** SEF (RS) has had breaking changes without notice. HR-FISK is in transition (Jan 2026 mandate). CPF (BA-FED) has no published tech specs. **Mitigation:** ADR-019 adapter lifecycle model (stub ? sandbox ? production).
2. **CoA regulatory changes:** Pravilnik updates require data migration, not schema migration. **Mitigation:** Versioned CoA table (ADR-017).
3. **BA political risk:** If BiH entities re-unify tax authorities, `BA_FED` and `BA_RS` would merge. **Mitigation:** Keep abstraction — consolidation is a data migration, not a code rewrite.

Implementation Notes

Plugin Selection (Runtime)

```kotlin
// In Ktor request context:
val org = jwt.claims["org"] as OrganizationClaims
val plugin = pluginRegistry.select(org.taxJurisdiction)
val vatResult = plugin.calculateVat(invoice)
```

No `if` branches in the core engine. The registry pattern ensures clean dispatch.

Validation Rules

1. `Organization.taxJurisdiction` is **NOT NULL** — enforced at DB level (check constraint).
2. Every MC task for Bilko must have `market:X` tag (Task 2.4 in master plan).
3. Data quality audit query runs nightly, alerts on un-tagged rows.

Testing Strategy

```kotlin
@Test
fun `RS plugin calculates PDV 20% correctly`() {
    val plugin = PluginRS()
    val invoice = CanonicalInvoice(
        lines = listOf(InvoiceLine(lineTotal = BigDecimal("100.0000"), taxRate =
BigDecimal("20.0000"))),
        jurisdiction = TaxJurisdiction.RS
    )
    val result = plugin.calculateVat(invoice)
    assertEquals(BigDecimal("20.0000"), result.vatAmount)
}
```

Each plugin has 20+ unit tests covering rate tiers, rounding, edge cases.

```
## References

- **Master plan:** `~/system/specs/bilko-multi-market-architecture-plan.md`
- **Implementation:** `~/ALAI/products/Bilko/scratch-
api/src/main/kotlin/no/alai/bilko/country/CountryPlugin.kt`
- **Related ADRs:**
  - ADR-016: E-Invoice Adapter & UBL 2.1 Canonical Model
  - ADR-017: RLS Multi-Tenancy Migration
  - ADR-018: Market vs Locale Separation
  - ADR-019: Integration Adapter Registry
- **Vendor research:**
  - Odoo l10n modules: https://github.com/odoo/odoo/tree/master/addons
  - Intuit Global-by-Design: https://www.intuit.com/blog/engineering/
  - Xero TaxEngine: https://developer.xero.com/documentation/api/accounting/taxrates

---

## Approval

**Approved:** 2026-04-21 by CEO Alem Basic
**Execution:** Phase 0 Task 0.1 (completed 2026-04-22, MC #8679–#8686)
```

Revision #2
Created 2026-04-22 10:36:56 UTC by John
Updated 2026-05-31 20:06:24 UTC by John