**🧠 Updated Conceptual Architecture: IPTV Data Harmonization System**

**🔹 1. Source Registry & Acquisition Layer**

Centralized config defines all M3U and EPG sources.

**Supports:**

* Static URLs, dynamic scripts, manual uploads
* Compression handling (.gz, .zip)
* Metadata: country, format, update frequency

**Example Config:**

sources:

- name: epg\_CA

type: epg

country: CA

url: https://epg.pw/xmltv/epg\_CA.xml.gz

format: xmltv

update: daily

decompress: true

- name: m3u\_US

type: m3u

country: US

url: https://freetv.fun/test\_channels\_united\_states\_new.m3u

format: m3u

update: weekly

**🔹 2. Normalization Layer**

All inputs converted to a canonical intermediate format.

**M3U → JSON:**

{

"channel\_id": "WXYZ",

"name": "ABC Detroit",

"group": "Entertainment",

"url": "http://stream.example.com/abc",

"country": "US",

"logo": "http://logo.example.com/abc.png"

}

**EPG → Unified XMLTV or JSON:**

{

"channel\_id": "WXYZ",

"programs": [

{ "start": "...", "end": "...", "title": "...", "desc": "...", "category": "News" }

]

}

**🔹 3. Alias Resolution & Matching Engine**

Matches M3U channels to EPG entries using:

* Alias dictionaries
* Fuzzy matching (e.g., CBLT-DT ↔ CBC Toronto)
* Manual overrides

**Naming Format:**

* US-ABC-WXYZ (Detroit)
* CA-CTV-CBLT (Toronto)
* US-HBO 1, UK-HBO 1

**Alias Config:**

aliases:

"CBLT-DT": { network: "CTV", station: "CBLT", location: "Toronto", country: "CA" }

"WXYZ": { network: "ABC", station: "WXYZ", location: "Detroit", country: "US" }

**🔹 4. Transformation Layer**

Applies user-defined rules to rename, regroup, filter, and format.

**Rule Types:**

* Rename channels
* Assign groups (CA Movie, US Movie)
* Filter unwanted entries
* Apply logos, sort order

**Rule Config:**

rules:

rename:

- match: "CBLT-DT"

replace: "CA-CTV-CBLT (Toronto)"

group:

- match: ".\*Movie.\*"

assign: "{country} Movie"

**🔹 5. Reconciliation & Exception Handling**

Handles:

* Unmatched channels
* Conflicts
* New or removed entries

**Output:**

* Match report (✅ matched, ❌ unmatched, ⚠️ conflicts)
* GUI dashboard for manual resolution

**🔹 6. Output Layer**

Generates:

* Final M3U file (transformed, grouped, sorted)
* Final EPG file (matched, cleaned)
* Optional: grouped M3Us per country/category

**Output Format:**

* Compatible with Tivimate and other IPTV apps
* Versioned and auditable

**🔹 7. GUI & Automation**

**GUI Modules:**

* Source Manager
* Mapping Dashboard
* Rule Editor
* Preview Panel
* Run History

**Automation:**

* Scheduled fetch + transform
* CLI for batch runs
* GUI for manual overrides

**🔹 8. External API Integration**

Use your API keys to enrich metadata:

* TMDB, OMDB, TVMaze, Trakt → enrich program info
* Google Geocoding → resolve locations
* OpenAI → suggest aliases or rules

**✅ Next Steps (Now Aligned)**

1. **Define canonical schema** for M3U and EPG intermediate formats
2. **Build source registry config** from your curated links
3. **Draft rule config** for naming, grouping, filtering
4. **Prototype alias resolution and fuzzy matching**
5. **Sketch GUI wireframes** for mapping and rule editing
6. **Extract reusable logic** from GitHub projects and your uploaded spec

**🧭 Master Plan: IPTV Data Harmonization System**

**🔹 Phase 0: Foundation Setup**

**Goal:** Establish the working environment and baseline assets.

* ✅ Organize all known M3U and EPG sources into a source registry
* ✅ Define naming conventions (e.g., US-ABC-WXYZ (Detroit))
* ✅ List all API keys and external enrichment services
* ✅ Upload and catalog all previous specs, POCs, and GitHub references

**🔹 Phase 1: Canonical Schema Design**

**Goal:** Define intermediate formats for M3U and EPG data.

* 📘 M3U → JSON schema with fields: channel\_id, name, group, url, country, logo
* 📘 EPG → Unified XMLTV or JSON schema with fields: channel\_id, start, end, title, desc, category
* 🧪 Create sample files for testing transformations

**🔹 Phase 2: Source Registry & Fetch Engine**

**Goal:** Build a config-driven fetcher for all data sources.

* 🧱 YAML config for each source (URL, format, country, update frequency)
* 🔁 Support for static URLs, dynamic scripts, manual uploads
* 🧼 Auto-decompression and staging of .gz, .zip, etc.
* 🧪 Validate fetch logic with a few known sources (e.g., epg\_CA, m3u\_US)

**🔹 Phase 3: Normalization & Alias Resolution**

**Goal:** Convert raw inputs to canonical format and resolve naming inconsistencies.

* 🧩 Apply regex, fuzzy matching, and alias dictionaries
* 🧠 Build override tables for ambiguous or conflicting names
* 🧪 Test with real-world examples (e.g., CBLT-DT, WABC, KCBS)

**🔹 Phase 4: Transformation Engine**

**Goal:** Apply user-defined rules to rename, regroup, filter, and format channels.

* 🛠️ YAML/JSON rule config for:
  + Renaming (CBLT-DT → CA-CTV-CBLT (Toronto))
  + Grouping (.\*Movie.\* → CA Movie)
  + Filtering (exclude Test Channel)
* 🧪 Apply rules to sample M3U/EPG pair and validate output

**🔹 Phase 5: Reconciliation & Matching**

**Goal:** Match M3U channels to EPG entries and handle exceptions.

* 🔍 Match by station code, network, country, fuzzy name
* ⚠️ Flag unmatched or conflicting entries
* 📊 Generate match report: ✅ matched, ❌ unmatched, ⚠️ conflicts
* 🧪 Manual override interface (CLI or GUI)

**🔹 Phase 6: Output Generation**

**Goal:** Produce clean, consistent M3U and EPG files for Tivimate.

* 📦 Final M3U with transformed entries
* 📦 Final EPG XMLTV file with matched programs
* 📦 Optional: grouped M3Us per country/category
* 🧪 Validate output in Tivimate or similar app

**🔹 Phase 7: GUI Design & Automation**

**Goal:** Build a user-friendly interface and automate the pipeline.

* 🖥️ GUI Modules:
  + Source Manager
  + Mapping Dashboard
  + Rule Editor
  + Preview Panel
  + Run History
* 🔁 Automation:
  + Scheduled fetch + transform
  + CLI for batch runs
  + GUI for manual overrides

**🔹 Phase 8: Metadata Enrichment**

**Goal:** Use external APIs to enhance channel and program data.

* 🌐 TMDB, OMDB, TVMaze, Trakt → enrich program info
* 🗺️ Google Geocoding → resolve location-based channels
* 🧠 OpenAI → suggest aliases, rules, or naming conventions

**✅ Immediate Next Steps**

Here’s what we’ll do first:

1. **Draft canonical schemas** for M3U and EPG intermediate formats
2. **Build source registry config** from your curated links
3. **Create rule config** for naming, grouping, filtering
4. **Prototype alias resolution logic** with fuzzy matching
5. **Test transformation on a sample M3U/EPG pair**

Once those are in place, we’ll move into GUI design and automation.