AccuFund → Mr. MoneyBags v1.x — Combined Migration Guide

(Step-by-Step + Reference)

1 Executive Summary

Mr. MoneyBags v1.x delivers a modern, web-based, multi-entity fund-accounting platform with integrated banking, robust reporting, and role-based security.
This guide merges the former *Migration Guide* and *Step-by-Step Checklist* into one concise document that:

- Describes the end-to-end migration flow from AccuFund to the current application.
- Provides actionable checklists, commands, and validation steps.
- Reflects the present data model (Account, Description, Classifications).

---

2 Scope & Assumptions

- Source system: AccuFund 9.x (on-prem).
- Target system: Mr. MoneyBags v1.x (current repo state).
- Field model updates
  - **Account (code)** — primary key (formerly *Account Code* label).
  - **Description** — replaces legacy *name*.
  - **Classifications** — replaces legacy *type* (Asset / Liability / …).
- Default deployment: Node 20+ & PostgreSQL 16+ on any OS.
  Containers are optional; instructions show non-container flow.

---

3 Pre-Migration Planning (condensed)

Role	Responsibility
Executive Sponsor	Approves timeline & budget
Finance Lead	Chart-of-Accounts & fund mapping
IT Lead	Environment setup, data extraction, validation
Training / Comms	User notices & training sessions

Checklist (□ = pending, ☑ = done)

- □ Inventory AccuFund modules & custom reports
- □ Identify integrations (bank feeds, payroll, CRM)
- □ Capture full & differential backups of AccuFund DB
- □ Schedule migration window and notify users
- □ Verify hardware / cloud resources for target stack

---

4 Environment Setup (current repo)

Prerequisites

- Node 20 or newer, npm 10+
- PostgreSQL 16+ (server & psql client)
- Git

Commands (≈ 5 min):

```
 git clone https://github.com/ORG/mr-moneybags-v1.x.git
cd mr-moneybags-v1.x
npm install                         # install dependencies
npm run setup                       # create schema + load sample data
npm run dev                         # runs API :3000 + client :8080
```

Optional utilities

- `npm run db:recreate` – drop & recreate database
- `npm run db:seed` – reload sample data only
- `npm run db:verify` – integrity checks

---

5 Data Extraction from AccuFund

Export	AccuFund Path	Output File
GL Accounts	GL → Export → Chart of Accounts	`gl_accounts.csv`
Funds	Funds → Export	`funds.csv`
Vendors	AP → Vendors	`vendors.csv`
Bank Accounts	Banking → Bank Accounts	`bank_accounts.csv`
Journal Detail (per FY)	Reports → GL Detail	`je_YYYY.csv`
Outstanding Checks	AP → Checks	`open_checks.csv`
Open Deposits	Cash Receipt → Deposits	`open_deposits.csv`

Save all files in `migration_exports/YYYY-MM-DD/`.

---

6 Mapping & Data Model Updates

- **COA Mapping Sheet** (create or use `templates/coa-mapping.xlsx`)

Old Field	New Field	Notes
Account #	Account (code)	May truncate/format
Name	**Description**	Required
Type	**Classifications**	Asset / Revenue / …

- **Fund Types** remain unchanged (Unrestricted / Temp. Restricted / Perm. Restricted).
- **Users & Roles**
  - Administrator → `admin`
  - General Ledger → `finance`
  - View Only → `viewer`

---

7 Staging & Import Workflow

1. **Load CSVs** into staging schema (use `\copy` or `/api/*/bulk-import`).
2. **Validate**
   - Record counts match exports
   - Required fields not null
   - Balanced journal entry totals
3. **Promote** — run `CALL sp_promote_staging_to_prod();` or POST `/api/migrate/promote`.
4. **Rebuild Balances** — `CALL sp_rebuild_balances();` to refresh aggregates.

Common validations

- Account codes unique per entity
- Funds linked to valid entities
- Debits = Credits on each JE

---

8 Banking Modules Setup

- **Bank Accounts** → Settings › Bank Accounts › Import CSV.
- **Statements Upload** → Bank Reconciliation › Statements › Upload (.OFX/.CSV).
- **Check Formats** → Check Printing › Check Formats (align & save).
- **Deposits** → Bank Deposits › New Deposit → Add items → Post.

---

9 Authentication & Users

1. Create initial admin:

```
 npm run create-admin -- --user admin --email admin@example.org --
password 'TempP@ss1'
```

2. Bulk-import users (CSV: full_name,email,role).
3. Audit roles in **Settings › Users**; only `admin` sees **Settings** nav.

---

10 Reports & Verification

Standard Reports

- Trial Balance
- Fund Balance / Activity / Statement
- Funds Comparison

Advanced Tools

- **Custom Reports** builder
- **Natural Language Queries** (Ask Questions page)

---

11 Testing & Go-Live Checklist

Validation

- □ Record counts staging vs prod
- □ Trial balance per entity = AccuFund
- □ Bank reconciliation ending balance matches statement
- □ Random JE drill-downs (5 per month migrated)
- □ Role-based UI access correct

Go-Live

1. Freeze AccuFund data entry (read-only).
2. Export & import delta transactions (last 48 h).
3. Switch DNS / URL to new server.
4. Monitor API & DB logs for 48 h, resolve errors.
5. Decommission AccuFund (archive backups).

---

12 Appendix A — Useful Commands

Purpose	Command
Setup schema & sample data	`npm run setup`
Full DB rebuild	`npm run db:recreate`
Verify integrity	`npm run db:verify`
Start dev servers	`npm run dev`
Dump prod DB	`pg_dump -U postgres -Fc mrmoneybags > dump_$(date +%F).pgc`
Restore dump	`pg_restore -U postgres -d mrmoneybags dump.pgc`

13 Appendix B — Data Templates

If `templates/coa-mapping.xlsx` is present, use it.
Otherwise create a spreadsheet with columns:

1. `old_account_number`
2. `new_account_code`
3. `description`
4. `classifications` (Asset / Liability / Equity / Revenue / Expense)
5. `entity_code` (optional for multi-entity orgs)

> **Notes**
>
> - All UI labels now show **Account** (not "Account Code").
> - Documentation formerly referencing *name/type* now uses **Description / Classifications**.
> - Commands are OS-agnostic; adjust paths and service names as needed.
> - Containers remain supported (`docker compose up -d`) but are not required.
