# HATPro — Validation & Testing Manual

# HATPro — Validation & Testing Manual

\_Last updated: 2025-10-30\_

## 1) Background: Auto-validation with AJV

We use \*\*AJV (Another JSON Schema Validator)\*\* to ensure that all \*\*generated artifacts\*\* compile and are structurally valid.

### Commands

npm run validate:schemas # Compile all generated .schema.json (draft2020)

npm run validate:enums # Compile all generated enum JSONs

These checks do \*\*not\*\* execute example instances; they only verify that generated schemas/enums are syntactically valid and internally consistent.

---

## 2) Two kinds of tests

We distinguish between two complementary test types:

### A) Auto-verify (generated) tests

- Scope: \*\*Generated artifacts\*\* only

- What: AJV \_compilation\_ of schemas/enums

- Where: `packages/\*\*/json/schemas/\*\*`, `packages/\*\*/json/enums/\*\*`

- Trigger: `npm run validate:schemas`, `npm run validate:enums`

- Clean-up: Removed by `npm run clean`

### B) Developer-authored example tests

- Scope: \*\*Hand-crafted example instances\*\* that should \*\*pass\*\* or \*\*fail\*\* against matching schemas

- Where (never cleaned): `packages/\*\*/json/examples/\*\*`

- Optional versioning: `packages/\*\*/json/examples/v1/\*\*`, `.../v2/\*\*`, etc.

- Trigger: `npm run validate:examples` (and filters like `--version v1`, `--segment core`, `--only CurrencyAmount`)

- Clean-up: \*\*Never\*\* removed by `npm run clean`

\*\*Naming convention\*\*

- `\*.valid.json` → \*\*must pass\*\* validation

- `\*.invalid.<reason>.json` → \*\*must fail\*\* validation

\*\*Folder structure example\*\*

packages/core/json/

schemas/ # GENERATED (cleanable)

enums/ # GENERATED (cleanable)

examples/ # DEV-AUTHORED (never cleaned)

README.md

commonLib/

CurrencyAmount.valid.json

CurrencyAmount.invalid.amount.json

CurrencyAmount.invalid.currency.json

samples/ # AUTO-GENERATED SAMPLE DATA (cleanable)

---

## 3) Cleaning policy

Our `clean` script removes only \*\*generated\*\* outputs and \*\*auto samples\*\*:

{

"clean": "rimraf --glob \"packages/\*\*/json/schemas\" \"packages/\*\*/json/enums\" \"packages/\*\*/json/samples\""

}

> It \*\*does not\*\* touch `json/examples/\*\*`.

---

## 4) Validating developer-authored examples

We ship a validator script that loads every example, resolves the matching schema, and asserts \*\*pass/fail\*\* according to filename.

### Commands

# All examples

npm run validate:examples

# Specific version (e.g., v1, v2) under json/examples/<version>/...

npm run validate:examples -- --version v1

# Limit to a package segment (e.g., core, travelProfile)

npm run validate:examples -- --segment core

# Only validate a particular type name (base class), e.g., CurrencyAmount

npm run validate:examples -- --only CurrencyAmount

### Exit behavior

- Exits \*\*0\*\* when all assertions succeed

- Exits \*\*1\*\* if any `\*.valid.json` fails or any `\*.invalid.\*.json` unexpectedly passes

---

## 5) CI recommendations

Use a fast, deterministic setup:

npm ci

npm run build:ci # strict lint, gen, and schema/enum validation

npm run validate:examples # assert example pass/fail

---

## 6) Troubleshooting quick tips

- \*\*Schema not found\*\*: check path mapping from `examples/.../<Name>.\*.json` → `schemas/.../<Name>.schema.json`

- \*\*Unexpected pass/fail\*\*: ensure filename contains `.valid.` or `.invalid.` exactly once

- \*\*Windows globbing\*\*: always quote globs; rely on `rimraf --glob` for portability

- \*\*Versioned examples\*\*: put them under `json/examples/vN/\*\*` and pass `--version vN`

---

## 7) Appendix — Suggested package.json scripts

{

"validate:schemas": "ajv compile -s \"packages/\*\*/json/schemas/\*\*/\*.schema.json\" --spec=draft2020",

"validate:enums": "ajv compile -s \"packages/\*\*/json/enums/\*\*/\*.json\" --spec=draft2020",

"validate:examples": "node tools/validate-examples.mjs",

"test": "npm run validate:enums && npm run validate:schemas && npm run validate:examples"

}

> Dev tools you may need: `ajv`, `ajv-cli`, `ajv-formats`, `ajv-errors`, `globby`, `rimraf`.