

OSF — Cadre Méthodique (B)

Transobserver — collecteur structuré (engines + schema + tools)

0) Identité / périmètre

- **Nom recommandé OSF (Title)** : *transobserver — collecte/structuration de métriques (engines + schema)*
- **Dépôt source (URL)** : github.com/dalozedidier-dot/transobserver
- **Type de ressource (Resource Type)** : *Software*
- **Statut observé** : dépôt public, 27 commits, langages détectés (Python/HTML/Shell/Jupyter).
- **Version à figer (Registration snapshot)** :
 - Branch : [main](#)
 - Tag : [\[...\]](#)
 - Commit hash : [\[...\]](#)
 - Date de figement : [\[...\]](#)

1) Définition (Description / Abstract)

transobserver est une couche logicielle de **collecte, structuration et préparation** de données de métriques. L'architecture est organisée autour de répertoires de **configuration, engines** (moteurs), **schema** (contrat de structure), **tools, examples, fixtures** et **test_data**, avec packaging via [pyproject.toml](https://pypi.org/project/pyproject.toml). Le produit attendu est un ensemble de **métriques structurées et versionnables**, destinées à l'audit et à l'exploitation en aval (sans interprétation intégrée au collecteur).

2) Matériels / composants à archiver (Materials)

Inclure dans l'OSF (copie/archive) :

- Répertoires :
 - `config/`, `engines/`, `schema/`, `tools/`, `examples/`,
`fixtures/prepared_bands/`, `test_data/`, `transobserver/`
- Fichiers :
 - `pyproject.toml`, `QUICKSTART.txt`, `README.fr.md`,
`TEST_DATA_LINKS.md`
- **Archive figée** : ZIP du dépôt au commit/tag enregistré (OSF “Files”).

3) Données (Data / Inputs)

- **Sources** : à déclarer (non déductible uniquement via listing UI) :
 - type de sources (logs, exports, API, fichiers locaux, etc.)
 - mode d'accès (pull/push), prérequis (tokens **non** stockés), granularité
- **Contrat** :
 - `schema/` est l'artefact central : il définit les structures attendues.
- **Références externes** :
 - si `TEST_DATA_LINKS.md` pointe des ressources hors dépôt, OSF doit contenir soit un snapshot, soit une référence versionnée (URL + date + hash si disponible).

4) Procédure opératoire (Methods)

Procédure minimale à décrire (opératoire, sans hypothèse implicite) :

1. Sélection d'une configuration dans `config/`.
2. Exécution d'un moteur dans `engines/` sur une source déclarée.
3. Production d'artefacts de métriques structurées conformes au `schema/`.
4. (Si implémenté) validation de conformité + génération de fixtures dans `fixtures/` / `test_data/`.

5. Archivage des outputs + logs d'exécution.

Commande(s) exacte(s) : à renseigner dans OSF depuis `QUICKSTART.txt` / scripts (non extractible de manière fiable ici sans lecture exécutable).

5) Sorties / produits (Outputs)

À figer dans OSF (au moins un exemple complet) :

- 1 output “métriques brutes structurées” conforme au schéma (format à préciser : CSV/JSON/NDJSON...)
- 1 output “fixtures/prepared_bands” (si ces bandes sont un standard interne d'exploitation)
- 1 jeu `test_data/` minimal pour non-régression
- Schéma (`schema/`) versionné et inclus.

6) Validation / critères de vérifiabilité (QA / Validation)

- Validation minimale :
 - conformité des outputs au `schema/`
 - reproductibilité sur `test_data/` (diff stable / snapshots)
- Si CI existe dans `.github/workflows/` : inclure la description des checks et les critères de succès.

7) Packaging / environnement (Reproducibility)

- `pyproject.toml` présent (packaging).
- À figer dans OSF :
 - version Python cible
 - dépendances effectives (lock/requirements ou export)
 - OS/runner si déterminant

8) Licence / réutilisation (License)

- **Champ OSF License** : à renseigner explicitement (même si dépôt sans LICENSE).

9) Limites (Limitations)

- Sans hash/tag + archive, l'artefact logiciel n'est pas fixable.
- Sans exemple d'output conforme + schéma inclus, l'outil reste non-auditable par tiers.
- transobserver ne conclut pas : il structure/collecte, il ne "décide" pas.

Note OSF (structure du dépôt OSF + registration)

- Un **Registration** OSF est un **snapshot immuable** ; c'est l'outil adapté pour figer un état logiciel + artefacts.
- Pour une registration "logiciel", le point non-compensable est : **archive figée + hash + licence + procédure + exemples entrée/sortie**.