# TIDeS Handbook — Chapter 0

## Governance, Versioning & Packaging (Normative Chapter)

**Audience.** Editors, maintainers, vendors, clinical sites, and researchers implementing or extending TIDeS.

**Outcome.** By the end of this chapter you will (a) stand up a standards‑compliant repository, (b) publish versioned releases with signed artifacts and DOIs, (c) operate an RFC process, and (d) prove conformance via CI and badges.

**Normative keywords:** **MUST**, **SHOULD**, **MAY** (RFC 2119/RFC 8174).

### 0.0 Why This Chapter Exists

TIDeS is executable: its rules are enforced by code and proven by fixtures. Execution requires predictable governance (who decides), careful versioning (what changed), and disciplined packaging (what ships). This chapter is the operating manual for those three pillars. Treat it like your Standard Operating Procedure (SOP).

## 0.1 Core Concepts & Boundaries

**Specification family.** TIDeS consists of: - **Spec text** (this Handbook/IG) — human‑readable normative requirements. - **Schemas** — machine‑readable constraints (JSON Schema 2020‑12). - **Validator** — CLI/API and rule pack that enforce the spec. - **Profiles** — A (Clinical‑Full), B (Research‑Voxel), C (Legacy‑Organ). - **Interops** — DICOM/FHIR/OMOP bindings. - **Examples/Fixtures** — canonical test bundles with golden reports. - **OMOP DDL** — relational export. - **Docs site** — publishable copy of the IG.

**Conformance surface.** Every artifact **MUST** expose its specVersion and include provenance (software, version, inputs, hash). If your artifact cannot declare both, it is not TIDeS‑conformant.

**Licensing split.** Code (schemas, validator) → **Apache‑2.0**. Narrative text (spec/docs) → **CC‑BY‑4.0**. Example data PHI‑free.

## 0.2 Repository Blueprint (Authoritative Layout)

Create a public repo with the following canonical structure. Deviations **MUST** be justified in GOVERNANCE.md.

TIDES/  
 spec/ # Normative narrative (this handbook)  
 schemas/ # JSON Schema 2020‑12  
 validator/ # CLI/API, rules, tests, container  
 api/ # OpenAPI + server  
 rules/expanded.json # Machine rules with spec clause pointers  
 tests/ # Unit + fixture tests  
 policy/ # Versioned OAR & FLASH policies  
 interops/  
 dicom/ # Tag lists, examples, policies  
 fhir/ # StructureDefinitions, examples  
 omop/ # Mapping docs  
 omop/  
 tides\_ddl.sql # OMOP DDL extensions  
 profiles/ # A|B|C requirements (machine + text)  
 examples/  
 case\_\* / bundle.json, report.json, report.txt, report.html, README.md  
 docs/ # mkdocs site (built per release)  
 rfcs/ # RFC proposals and history  
 .github/workflows/  
 validate.yml # CI: lint, test, fixtures, docs  
 release.yml # Release: build, sign, publish, DOI  
 CITATION.cff  
 LICENSES/  
 LICENSE-Apache-2.0  
 LICENSE-CC-BY-4.0  
 SECURITY.md  
 CODE\_OF\_CONDUCT.md  
 GOVERNANCE.md  
 CONTRIBUTING.md  
 RELEASE\_NOTES.md  
 MANIFEST.json # Checksums + signatures  
 spec/traceability.csv # Clause → RuleID → Schema → Fixtures

**Naming rules (normative).** - Schemas **MUST** be named tides-<kebab>.schema.json and carry $id that pins the spec version (https://tides.org/schemas/1.0.0/...). - Fixtures **MUST** be case\_\* directories with an explanatory README.md and expected outcomes.

## 0.3 Roles & Decision‑Making (Governance)

### 0.3.1 Bodies & Responsibilities

* **Steering Committee (SC)** — roadmap, MAJOR/MINOR approval, dispute resolution.
* **Editors** — spec/handbook and profiles; merge rights over /spec, /profiles.
* **Maintainers** — schemas, validator, rule pack, CI.
* **Implementer Advisory Group (IAG)** — vendors/sites; non‑binding guidance; monthly calls.

### 0.3.2 Quorum, Minutes, and Transparency

* SC decisions require a simple majority; votes and rationales **MUST** be recorded under /spec/minutes/.
* Editor and Maintainer meetings **SHOULD** publish notes within 7 days.

### 0.3.3 Terms & Conduct

* 1‑year renewable terms; selection documented in GOVERNANCE.md.
* CODE\_OF\_CONDUCT.md applies to all contributors and meetings.

## 0.4 Versioning Policy (SemVer)

### 0.4.1 Spec SemVer

* **MAJOR**: breaking changes to schemas or rule semantics → 2.0.0.
* **MINOR**: additive, non‑breaking features; new WARN rules; optional fields → 1.1.0.
* **PATCH**: clarifications, typos, docs, test fixes → 1.0.1.
* Every artifact **MUST** declare specVersion equal to the tag.

### 0.4.2 Profile Coupling

Profiles A/B/C carry their own minor counters (A.1, B.1, C.1) and **MUST** align to a single spec MAJOR.MINOR (e.g., 1.0.x). Raising a profile requirement to **MUST** demands at least a spec MINOR.

### 0.4.3 Deprecations

* Mark items **DEPRECATED** in a MINOR; continue accepting for **two subsequent MINORs** within the MAJOR; remove at next MAJOR.
* Validator emits INFO:DEPRECATION with sunsetAt metadata.

### 0.4.4 Policy Packs (Safety/FLASH)

* Version policy bundles independently (policy-oar-1.0.0, policy-flash-1.0.0).
* Each pack declares specMajor: 1; validator rejects mismatches.

## 0.5 RFC Workflow (Change Control)

**Lifecycle:** proposal → accepted → released (or rejected/superseded).

**Mandatory sections:** Motivation, alternatives, back‑compat, security/privacy, migration, test/fixture updates, rule changes.

**Acceptance gate:** Two Editor approvals + Maintainer sign‑off (validator impact). SC approval required for MAJOR/MINOR.

**Traceability:** Upon acceptance, reference RFC in RELEASE\_NOTES.md and annotate affected rules with specClause and rfc fields in rules/expanded.json.

**Starter template** lives at rfcs/RFC-TEMPLATE.md.

## 0.6 Packaging, Signing, and Publication

### 0.6.1 Release Channels

* **Stable** (default), **Next** (e.g., 1.1.0‑rc.1), **LTS** (annual; 18‑month bugfix window).

### 0.6.2 Tagging & Artifacts (Normative)

Tag vMAJOR.MINOR.PATCH and attach: 1. tides-spec-<ver>.zip (spec + docs source) 2. tides-schemas-<ver>.zip 3. tides-validator-cli-<ver>.tar.gz (wheel/sdist) 4. tides-validator-container-<ver>.txt (OCI digest) 5. tides-examples-<ver>.zip (fixtures + golden reports) 6. tides-omop-ddl-<ver>.sql 7. MANIFEST.json (SHA‑256 + signatures)

### 0.6.3 DOIs & Citations

* Archive releases (e.g., Zenodo) and embed DOI in CITATION.cff and docs home.
* Keep contributor list current.

### 0.6.4 SBOM & Signatures

* Produce **CycloneDX** SBOMs for validator CLI/container.
* Sign OCI images (cosign). Publish public keys in SECURITY.md.

**MANIFEST.json (pattern)**

{  
 "version": "1.0.0",  
 "specVersion": "1.0.0",  
 "artifacts": [  
 {"name": "tides-schemas-1.0.0.zip", "sha256": "<hex>"}  
 ],  
 "signing": {"cosign": "<sig-ref>"}  
}

## 0.7 Continuous Integration (Quality Gates)

Your CI **MUST**: 1. Lint and validate schemas. 2. Run validator unit tests — 100% of rule IDs referenced. 3. Execute all fixtures and diff against golden reports (JSON + text + HTML). 4. Verify profile MUST‑matrix coverage (A/B/C). 5. Build docs and run link check with --strict. 6. Generate SBOM and run dependency/container vulnerability scans.

**validate.yml (excerpt)** lives in .github/workflows/ and includes install, lint, tests, fixtures, docs.

## 0.8 Security & Privacy Operations

* **PHI**: Prohibited in fixtures/examples. Any real‑world traces **MUST** be de‑identified.
* **Telemetry**: Off by default. If present, --telemetry flag **MUST** anonymize counts only.
* **Disclosure**: Report vulnerabilities privately per SECURITY.md. Aim for 90‑day coordinated disclosure.

## 0.9 Provenance & Reproducibility

Every produced report (e.g., validator output) **MUST** embed:

{  
 "software": "tides-cli",  
 "version": "1.0.0",  
 "commit": "<git-sha>",  
 "containerDigest": "sha256:<digest>",  
 "buildDate": "<ISO8601>"  
}

Reproducible builds **SHOULD** be documented (CONTRIBUTING.md).

## 0.10 Badging & Evidence of Compliance

On full success the validator emits a badge JSON:

{  
 "profile": "A",  
 "result": "PASS",  
 "specVersion": "1.0.0",  
 "validator": "1.0.0",  
 "timestamp": "2025-09-25T07:00:00Z",  
 "inputSha256": "<hex>"  
}

Publish this alongside manuscripts, registries, or data releases.

## 0.11 Traceability Matrix (Living Contract)

Maintain /spec/traceability.csv mapping **Clause → RuleID → Schema path → Fixture(s)**. Use it to assert that every normative statement is enforced, and every rule is justified by text.

**Example row**

0.4.1,spec-version,/specVersion,case\_pass\_minimal

## 0.12 Hands‑On: Standing Up TIDeS in One Day

**Goal.** Initialize a repo, wire CI, publish a signed pre‑release.

1. **Scaffold** the repo using the blueprint in §0.2.
2. **Insert** minimal schemas: tides-timings.schema.json and tides-dose-report.schema.json with version‑pinned $id and specVersion.
3. **Implement** a thin validator CLI with 3 core rules (spec-version, timing-injection, unit-ucum).
4. **Add** three fixtures: case\_pass\_minimal, case\_fail\_units, case\_provenance\_missing with golden reports.
5. **Wire** .github/workflows/validate.yml and make CI green.
6. **Tag** v1.0.0-rc.1, generate MANIFEST.json, **sign** the container, attach artifacts.
7. **Archive** the release and obtain a DOI; update CITATION.cff.

If any step fails, file an issue tagged with S1-critical and reference this chapter.

## 0.13 Checklists & Templates

### Release Readiness (TIDeS‑CHK‑0)

* specVersion synced across all artifacts
* Fixtures pass with expected statuses
* rules/expanded.json ↔ traceability.csv aligned
* SBOMs generated; MANIFEST.json complete
* DOI minted; CITATION.cff updated
* Docs site builds with zero link errors
* Security scans show no blocking CVEs

### RFC Cover Sheet

RFC‑XXXX — <title>  
Impact: MAJOR | MINOR | PATCH  
Summary:  
Motivation:  
Alternatives considered:  
Backward compatibility:  
Security/Privacy impact:  
Migration plan:  
Test plan + fixtures:  
Rule changes (IDs, severities):

### CITATION.cff Skeleton

cff-version: 1.2.0  
message: "If you use TIDeS, please cite this release."  
title: "TIDeS 1.0 — Theranostics Interoperability, Dosimetry & Safety"  
version: 1.0.0  
doi: 10.5281/zenodo.XXXXXXX  
authors:  
 - family-names: "TIDeS Steering Committee"  
 given-names: "Editors"  
date-released: 2025-09-25  
repository-code: https://github.com/<org>/tides  
license: CC-BY-4.0

## 0.14 Frequently Asked Questions (Operational)

**Q: Can we change directory names?**  
A: Only with justification in GOVERNANCE.md. Tooling assumes the canonical layout.

**Q: How do we handle local site policies (e.g., FLASH thresholds)?**  
A: Ship as versioned policy packs (§0.4.4). Validator refuses incompatible specMajor.

**Q: Our site cannot publish DOIs. What now?**  
A: The *spec* release DOI is mandatory for the standards body; implementers **SHOULD** cite that DOI in lieu of a site DOI.

**Q: Can we embed telemetry?**  
A: Only opt‑in; default off. Never transmit PHI.

## 0.15 Non‑Conformance & Remediation

When CI fails: 1. **Classify** severity (S0–S3). 2. **Open** an issue with failing logs and the rule IDs. 3. **Patch** tests/fixtures first; then rules; then schemas; finally text. Avoid changing the text to match bugs. 4. **Backport** to LTS if applicable.

## 0.16 Minimal Normative Snippets (For Copy‑Paste)

**Schema header with $id and pinned specVersion**

{  
 "$id": "https://tides.org/schemas/1.0.0/tides-dose-report.schema.json",  
 "$schema": "https://json-schema.org/draft/2020-12/schema",  
 "title": "TIDeS Dose Report",  
 "type": "object",  
 "required": ["id", "specVersion", "subject", "series"],  
 "properties": {  
 "specVersion": {"type": "string", "const": "1.0.0"}  
 }  
}

**Validator API contract (header pinning)** - Clients **MUST** send X-TIDeS-Spec: 1.0.0 to /validate. - Server **MUST** reject MAJOR mismatches with 409 Conflict.

## 0.17 Chapter Summary

* Governance defines *who decides*; SemVer defines *how change lands*; packaging defines *what ships*.
* Evidence comes from fixtures and badges, not opinions.
* If it isn’t versioned, signed, and reproducible, it isn’t TIDeS.

### Appendix 0‑A: Glossary (Chapter‑Local)

**SC** — Steering Committee.  
**IAG** — Implementer Advisory Group.  
**SBOM** — Software Bill of Materials (CycloneDX).  
**LTS** — Long‑Term Support release line.  
**Fixture** — Canonical example with expected validator result.  
**Badge** — Machine‑readable PASS attestation emitted by validator.

# TIDeS Handbook — Chapter 1

## Axioms & Definitions (Normative Chapter)

**Purpose.** This chapter defines the **first principles** of TIDeS—the irreducible rules and precise meanings that make the rest of the standard computable. If you get Chapter 1 right, everything downstream (schemas, validator, dosimetry, outcomes) becomes predictable and reproducible.

**Audience.** Clinical physicists, nuclear medicine physicians, imaging scientists, software engineers, data stewards, and regulators.

**Outcome.** By the end you will (a) use the same units and clocks, (b) describe geometry identically, (c) define dose‑rate and FLASH the same way at every site, (d) quantify uncertainty rigorously, and (e) attach provenance that proves what you did.

**Normative keywords:** **MUST**, **SHOULD**, **MAY** (RFC 2119/RFC 8174).

## 1.0 The Five Axioms (Normative)

1. **Units are UCUM.** All quantitative values **MUST** use SI/UCUM units with explicit symbols (e.g., Gy, Gy/s, Bq, s). No implicit prefixes, no unitless dose.
2. **Geometry is anchored by FrameOfReferenceUID (FoR).** Every spatial dataset **MUST** be tied to a DICOM **FoR**; any resampling **MUST** declare the transform used.
3. **Time is explicit and monotonic.** The injection start is an absolute ISO‑8601 timestamp; acquisition frames are explicit {tMid\_s, duration\_s} relative to injection; model windows are declared.
4. **FLASH is policy‑driven, voxel‑evaluated.** A voxel’s FLASH flag is determined by a versioned, site‑selectable policy; the **default** policy is provided here and **MUST** be used unless a versioned override is declared.
5. **Every number must be explainable (provenance & uncertainty).** Derived values **MUST** declare method and software provenance; key metrics **MUST** include uncertainty estimates with stated method (Type A/Type B; delta‑method or bootstrap).

**Validator hooks.** The validator **MUST** enforce these axioms via rule IDs: - unit-ucum, unit-dose-gy, unit-doserate - registration-fuid, registration-transform-provenance - timing-injection, timing-frames, timing-fitwindow - flash-policy-version, flash-coverage - provenance, uncertainty-declared

## 1.1 Units & Measurement System

### 1.1.1 Allowed Units (UCUM)

All quantitative fields **MUST** use UCUM tokens. The following table is **normative** for core quantities.

| Quantity | Symbol (UCUM) | Description | Examples |
| --- | --- | --- | --- |
| Activity | Bq | decays per second | 3.2e7 Bq |
| Time‑integrated activity (TIA) | Bq·s | (A(t),dt) | 1.8e10 Bq·s |
| Absorbed dose | Gy | J/kg | 18.5 Gy |
| Dose‑rate | Gy/s | time derivative of dose | 45 Gy/s |
| Time | s, min, h | seconds preferred | tMid\_s=2700 |
| Mass density | g/cm3 | from HU→ρ mapping | 1.04 g/cm3 |

**Prohibitions (normative).** - mGy for absorbed dose in **reports** is **NOT** permitted; convert to Gy. - Non‑UCUM strings (e.g., Gy/s^-1) are invalid. - Implied prefixes (e.g., Bq where MBq was meant) are prohibited.

**Anti‑patterns & fixes.** - *Anti‑pattern:* "dose": 18500, "unit": "mGy" → *Fix:* "dose\_Gy": 18.5. - *Anti‑pattern:* "rate": "45 Gy s-1" → *Fix:* "doseRate\_Gy\_per\_s": 45.

**Validator rules:** unit-ucum, unit-dose-gy, unit-doserate (ERROR).

## 1.2 Geometry, Registration & Coordinate Integrity

### 1.2.1 Frame of Reference (FoR)

* Every volumetric artifact (images, SEG, RTDOSE, dose maps, parametric maps) **MUST** declare the DICOM **FrameOfReferenceUID** it inhabits.
* A derived grid (e.g., resampled dose) **MUST** also carry parentFoR and a pointer to the transform object used.

**Minimum spatial header (normative JSON excerpt):**

{  
 "geometry": {  
 "frameOfReferenceUID": "1.2.826.0.1.3680043.2.1125.1.1234.5678",  
 "grid": {"spacing\_mm": [2.0, 2.0, 2.0], "size": [256, 256, 160]},  
 "registration": {  
 "type": "deformable",  
 "ref": "dicom:1.2.840.10008.5.1.4.1.1.66.4",  
 "provenance": {"software": "tides-reg", "version": "1.0.0", "hash": "<sha>"}  
 }  
 }  
}

### 1.2.2 Transform Provenance

* Rigid/affine/deformable transforms **MUST** be stored (DICOM Spatial Registration IOD) and referenced by UID.
* Resampling **MUST NOT** discard or overwrite transform provenance.

**Validator rules:** registration-fuid (ERROR), registration-transform-provenance (ERROR).

## 1.3 Time Semantics & Sampling Adequacy

### 1.3.1 Injection & Frame Model

* injectionStart **MUST** be ISO‑8601 with timezone (Z or offset). Example: 2025-09-25T11:03:27Z.
* frames **SHOULD** enumerate acquisitions with { tMid\_s, duration\_s } relative to injectionStart.
* fitWindow\_s **MAY** declare [t0, t1] for model fitting (seconds since injection).

**Normative schema skeleton:** see Chapter 8 schemas; enforced here by rule pointers.

**Examples (valid):**

{  
 "injectionStart": "2025-09-25T11:03:27Z",  
 "frames": [  
 {"tMid\_s": 600, "duration\_s": 120},  
 {"tMid\_s": 3600, "duration\_s": 180},  
 {"tMid\_s": 21600, "duration\_s": 240}  
 ],  
 "fitWindow\_s": [0, 86400]  
}

**Sampling adequacy (normative WARNs).** - Mono‑exponential PK **SHOULD** have **≥3** timepoints (early, mid, late). - Bi‑exponential PK **SHOULD** have **≥4** timepoints (at least one early & two post‑peak). - The validator **MUST** emit sampling-adequacy WARN when minima are not met.

**Edge cases.** If dynamic PET lists sub‑frame bins, you **MAY** aggregate to analysis frames but **MUST** preserve original bins under rawFrames for auditability.

**Validator rules:** timing-injection (ERROR), timing-frames (ERROR), sampling-adequacy (WARN).

## 1.4 FLASH Dose‑Rate Semantics

### 1.4.1 Default Policy (Normative)

A voxel is tagged flash=true when **instantaneous dose‑rate ≥ 40 Gy/s for ≥ 1 ms**. Sites **MAY** override this via a versioned policy pack; if so, the pack **MUST** be declared in provenance (see §1.6) and must specify thresholds and integration time.

**Formal definition.** Let ( (,t) ) be voxel dose‑rate at location (). Define a window (t = 1,). Then [ flash() =

]

**Reporting requirements.** - flashPolicy: { name, version, thresholds: { doseRate\_Gy\_per\_s, window\_ms } }. - flashCoverage\_pc per ROI and globally. - If policy is overridden, include policySourceURI and specMajor.

**Validator rules:** flash-policy-version (ERROR), unit-doserate (ERROR), flash-coverage (WARN if missing coverage metrics).

**Note (non‑normative):** The biophysical basis and clinical efficacy of FLASH are outside this spec; TIDeS standardizes *computation and reporting* only.

## 1.5 Uncertainty — Types, Methods, and Reporting

### 1.5.1 Taxonomy

* **Type A** (statistical): repeatability/fit variance derived from the data (e.g., WLS covariance of PK parameters, bootstrap of time‑activity curves).
* **Type B** (systematic): calibration factors, cross‑calibration to dose calibrator, partial‑volume corrections, kernel selection.

### 1.5.2 Required Reporting (Normative)

For every **key metric** (organ Dmean, voxel Dmax summaries, TIA per ROI), you **MUST** report: - **uncertainty\_pc**: total percent uncertainty (1‑σ unless otherwise specified). - **uncertaintyMethod**: delta-method | bootstrap | vendor-analytic | other (with reference). - **uncertaintyBreakdown** (SHOULD): { typeA\_pc, typeB\_pc } with method notes.

**Example (YAML excerpt):**

organSummaries:  
 - roiCode: SNOMED:74281007  
 roiCodeText: liver  
 Dmean\_Gy: 18.2  
 uncertainty\_pc: 9.1  
 uncertaintyMethod: delta-method  
 uncertaintyBreakdown: { typeA\_pc: 6.5, typeB\_pc: 6.6 }

### 1.5.3 Methods (Informative but Testable)

* **Delta‑method:** Propagate parameter covariance of PK fit through TIA and dose convolution. The covariance **MUST** be computed from the actual weighted fit used.
* **Bootstrap:** Resample time‑activity points (with replacement), refit PK, recompute TIA and dose for **≥1000** iterations (SHOULD), then report the 68% interval half‑width as 1‑σ.

**Validator rules:** uncertainty-declared (ERROR if missing fields), uncertainty-method-known (WARN on unknown method).

## 1.6 Provenance — Make Every Result Explainable

### 1.6.1 Minimum Provenance Block (Normative)

Every derived artifact **MUST** include the following provenance object:

{  
 "software": "tides-cli",  
 "version": "1.0.0",  
 "hash": "sha256:<artifact-bundle>",  
 "inputs": ["dicom:...", "seg:...", "rtDose:..."],  
 "params": {"model": "monoexp", "kernel": "S-value-177Lu-v1"},  
 "policyPacks": [  
 {"name": "policy-oar", "version": "1.0.0", "specMajor": 1},  
 {"name": "policy-flash", "version": "1.0.0", "specMajor": 1}  
 ],  
 "build": {"commit": "<git-sha>", "containerDigest": "sha256:<digest>", "time": "2025-09-25T07:00:00Z"}  
}

### 1.6.2 Reproducibility Expectations

* If params or inputs differ, the output **MUST** differ in hash and be considered a new artifact (immutable derivations).
* If software version changes, the validator **SHOULD** emit an INFO noting potential numerical differences.

**Validator rules:** provenance (ERROR), policy-compat (ERROR if specMajor mismatch).

## 1.7 Core Mathematical Definitions (Normative)

### 1.7.1 Activity & TIA

* **Activity** (A(t)) is in Bq.
* **TIA** per voxel or ROI is: [ = \_{0}^{} A(t),dt] Computed as trapezoidal integration over measured frames plus analytic tail from the chosen PK model (see Chapter 5).

### 1.7.2 Absorbed Dose & Dose‑Rate

* **Absorbed dose** (D()) in Gy is computed by convolution of time‑integrated energy deposition kernels with (). For S‑value formalism: [ D() = \_{s} S\_s(s), (s) ]
* **Dose‑rate** ((,t)) in Gy/s is the time derivative of dose.

### 1.7.3 Registration & Resampling Operators

Let (T: 33) be the mapping from source to target FoR. Resampling operator (R\_T) **MUST** be declared (nearest, linear, B‑spline) and linked to the transform instance.

**Validator rules:** kernel-metadata-declared (WARN in Chapter 5), registration-fuid (ERROR).

## 1.8 Coding Systems & Text Standards

* **ROIs** **MUST** carry codes from **SNOMED CT** or **RadLex**; include human‑readable roiCodeText.
* **Outcomes** **SHOULD** use **CTCAE** for toxicity and **iRECIST/PERCIST‑ext** for response.
* **Units** **MUST** be **UCUM**.

**Validator rules:** codesystem-roi (WARN if missing; ERROR for Profiles A/B where mandated by Chapter 3).

## 1.9 Examples (Canonical, Minimal)

### 1.9.1 Minimal TidesDoseReport (YAML)

id: urn:tides:dosereport:3b8c...  
specVersion: 1.0.0  
subject: urn:tides:subject:9a2f...  
series:  
 - doseMap: urn:tides:dosemap:f1d2...  
 organSummaries:  
 - roiCode: SNOMED:74281007  
 roiCodeText: liver  
 Dmean\_Gy: 18.2  
 Dmax\_Gy: 21.5  
 Vx: { "10Gy": 0.21 }  
 uncertainty\_pc: 9.1  
 flashCoverage\_pc: 4.3  
provenance:  
 software: tides-cli  
 version: 1.0.0  
 hash: 5f6e...  
 inputs: ["dicom:...", "seg:..."]  
 params: { model: monoexp }  
 policyPacks:  
 - { name: policy-flash, version: 1.0.0, specMajor: 1 }

### 1.9.2 Minimal Timing Block

{  
 "injectionStart": "2025-09-25T11:03:27+00:00",  
 "frames": [  
 {"tMid\_s": 300, "duration\_s": 60},  
 {"tMid\_s": 1800, "duration\_s": 120},  
 {"tMid\_s": 21600, "duration\_s": 240}  
 ],  
 "fitWindow\_s": [0, 86400]  
}

## 1.10 Pitfalls & How the Validator Catches Them

* **Unit drift** (MBq vs Bq): rule unit-ucum checks token set; schema pins allowed symbols.
* **Missing FoR** on dose maps: registration-fuid (ERROR) blocks Profile A/B.
* **Implicit clock**: absent timezone on injectionStart triggers timing-injection (ERROR).
* **Undeclared FLASH policy** for sites with custom thresholds: flash-policy-version (ERROR).
* **Uncertainty omitted**: uncertainty-declared (ERROR) for key metrics.

## 1.11 Checklists

**TIDeS‑CHK‑1 (Axioms Ready)** - [ ] All numeric fields use UCUM; absorbed dose in Gy; dose‑rate in Gy/s. - [ ] Every spatial artifact declares FoR; transforms stored and referenced. - [ ] injectionStart is ISO‑8601 with offset; frames listed with {tMid\_s, duration\_s}. - [ ] FLASH policy present (default or versioned override) and coverage computed. - [ ] Uncertainty fields present with method; provenance block complete.

## 1.12 Traceability Map (Clause → RuleID → Schema → Fixtures)

1.1,unit-ucum,/series[\*]/organSummaries[\*]/Dmean\_Gy,case\_fail\_units  
1.2,registration-fuid,/series[\*]/doseMap,case\_registration\_link\_missing  
1.3,timing-injection,/injectionStart,case\_pass\_minimal  
1.4,flash-policy-version,/provenance/policyPacks[\*],case\_flash\_flag\_coverage  
1.5,uncertainty-declared,/series[\*]/organSummaries[\*]/uncertainty\_pc,case\_pass\_minimal

## 1.13 Frequently Asked Questions (Focused)

**Q: Can I report dose in mGy?**  
A: Internally yes, but **reports** MUST be in Gy. Convert and round per Chapter 12 reporting rules.

**Q: Our scanner clock drifts. What then?**  
A: Record measured offsets in clockSync and either correct timestamps or document residuals; validator records INFO if drift noted.

**Q: Do I need voxel‑wise uncertainty?**  
A: Not required. Organ summaries **MUST** report uncertainty; voxel uncertainty is **MAY**.

## 1.14 Chapter Summary

* UCUM units, FoR‑anchored geometry, explicit time, policy‑driven FLASH, and mandatory provenance/uncertainty are the bedrock.
* These axioms do not constrain *how* you do science; they ensure that what you report is **computable, portable, and verifiable**.

### Appendix 1‑A: Symbol Table

* (A(t)) — activity [Bq]
* () — time‑integrated activity [Bq·s]
* (D) — absorbed dose [Gy]
* () — dose‑rate [Gy/s]
* **FoR** — DICOM FrameOfReferenceUID

### Appendix 1‑B: Example Policy Pack (FLASH)

name: policy-flash  
version: 1.0.0  
specMajor: 1  
thresholds:  
 doseRate\_Gy\_per\_s: 40  
 window\_ms: 1  
notes: "Default policy per TIDeS 1.0 §1.4"

# TIDeS Handbook — Chapter 2

## Information Model & Identifiers (Normative Chapter)

**Purpose.** Establish a precise, computable **information model** for theranostics workflows and define the **identifier system** that guarantees global uniqueness, immutability, and traceable derivations. This chapter is the ontology and wiring diagram for everything you will store, validate, exchange, and publish under TIDeS.

**Audience.** Solution architects, data modelers, EMR/VNA integrators, imaging/physics software vendors, registry designers, and regulators.

**Outcome.** You will (a) model theranostics studies using a fixed set of TIDeS resources, (b) assign stable, versioned identifiers, (c) maintain provenance‑linked derivation graphs, (d) serialize the model into JSON/YAML, DICOM, FHIR, and OMOP, and (e) pass profile A/B/C validation consistently.

**Normative keywords:** **MUST**, **SHOULD**, **MAY** (RFC 2119/RFC 8174).

## 2.0 Overview: Resource Graph (Authoritative)

TIDeS resources form a **directed acyclic graph (DAG)**. Parent → child edges represent derivations or containment. Every resource **MUST** carry id (URN), specVersion, and provenance (§1.6).

TidesStudy (root per clinical/research episode)  
 ├─ TidesImaging (one per acquisition series or timepoint set)  
 │ ├─ TidesSeg (DICOM SEG labelmaps per ROI set)  
 │ └─ TidesCalibration (site/scanner factors, cross-cal)  
 ├─ TidesPKModel (PK fit parameters per voxel/ROI)  
 ├─ TidesDoseMap (voxel dose, RTDOSE grid, per nuclide)  
 │ └─ TidesDoseReport (organ/ROI summaries, safety flags)  
 └─ TidesOutcome (response & toxicity over time)  
  
Auxiliary: TidesProvenance (shared provenance), PolicyPack references (OAR/FLASH).

**Validator rules (top‑level):** id-urn-format (ERROR), graph-dag (ERROR on cycles), provenance (ERROR), codesystem-roi (WARN/ERROR per profile), spec-version (ERROR).

## 2.1 Identifier System (Global URNs)

### 2.1.1 Format (Normative)

All TIDeS identifiers **MUST** be URNs in the namespace urn:tides: followed by a **type** and a **UUIDv4** (or ULID) component. Extended, human‑friendly segments **MAY** appear between type and UUID but **MUST NOT** be used for semantics.

urn:tides:<type>[:<flavor>]:<UUID>

**Allowed <type> values (normative):** study, imaging, seg, calibration, pkmodel, dosemap, dosereport, outcome, subject, policy.

**Examples:** - urn:tides:study:1b2f9c60-0e8a-4d6d-9e39-7c4e4d2f9a6a - urn:tides:pkmodel:monoexp:7a0d9c9e-0a71-4f79-9e22-57e7012b9c8b - urn:tides:dosemap:177Lu:3f7a7a2e-b9a5-4e93-8d0a-4b32f4e9d2bd

### 2.1.2 Requirements

* IDs **MUST** be immutable across the artifact lifetime.
* IDs **MUST NOT** encode PHI.
* Child references **MUST** use URNs, not file paths. External binary objects **MAY** appear as dereferenceable URIs in attachments[] but never replace the URN.
* When serializing to FHIR or OMOP, native identifiers **SHOULD** be preserved as identifier elements or surrogate keys.

**Validator rules:** id-namespace (ERROR if not urn:tides), id-uuid (ERROR), id-uniqueness (ERROR within bundle), id-referential (ERROR if dangling reference).

## 2.2 Resource Definitions (Normative Schemas)

All resources share id, specVersion, meta, and provenance. Profiles A/B/C set **Must‑Support** fields per §2.10.

### 2.2.1 TidesStudy

Represents a single theranostics episode (e.g., a cycle or protocol phase) for one subject.

**Cardinality & Fields (normative):** - id **MUST** — urn:tides:study:<UUID> - specVersion **MUST** — string (e.g., 1.0.0) - subject **MUST** — urn:tides:subject:<UUID> (PHI not embedded) - label **SHOULD** — free text (e.g., 177Lu-DOTATATE Cycle 1) - context **SHOULD** — { indication, nuclide, agent, protocolId, site } - timeframe **SHOULD** — { start, end } (ISO‑8601) - imaging[] **MAY** — array of TidesImaging URNs - policyPacks[] **SHOULD** — OAR/FLASH packs in use - attachments[] **MAY** — links to EMR orders or PDFs (non‑PHI if public)

**State machine (normative):** draft → in-progress → complete → amended (immutable id; amended creates new versionTag but retains same id only for minor clarifications that do not alter quantitative outputs; otherwise create a new id with provenance.parent).

**Example (YAML):**

id: urn:tides:study:1b2f9c60-0e8a-4d6d-9e39-7c4e4d2f9a6a  
specVersion: 1.0.0  
subject: urn:tides:subject:9a2f...  
context: { indication: NET, nuclide: 177Lu, agent: DOTATATE, protocolId: SSTR-CLN-01, site: JURA-NM }  
policyPacks: [ { name: policy-oar, version: 1.0.0, specMajor: 1 }, { name: policy-flash, version: 1.0.0, specMajor: 1 } ]

### 2.2.2 TidesImaging

Describes an acquisition cohort (one or more timepoints of PET/NM/CT/MR tied to a single FoR series or registered set).

**Fields:** - id **MUST** — urn:tides:imaging:<UUID> - dicomSeries[] **MUST** — list of SOP/SeriesInstanceUID references or URIs to VNA objects - timings **MUST** — per Chapter 8 tides-timings.schema.json - frameOfReferenceUID **MUST** — DICOM FoR - realWorldValueMaps[] **SHOULD** — units/scales for SUV, Ki, TIA maps - calibration **SHOULD** — TidesCalibration URN - seg[] **MAY** — zero or more TidesSeg URNs

**Constraints:** all referenced images **MUST** be either in a common FoR or have registered transforms.

### 2.2.3 TidesSeg

Semantic segmentation for ROIs with coded meanings.

**Fields:** - id **MUST** — urn:tides:seg:<UUID> - dicomSeg **MUST** — reference to DICOM SEG object(s) - roi[] **MUST** — list of { code, codeSystem, text, algorithm } - frameOfReferenceUID **MUST** — matches parent imaging FoR or carries transform

**Constraints:** For Profiles A/B, roi[].codeSystem **MUST** be SNOMED CT or RadLex.

### 2.2.4 TidesCalibration

Scanner & dose‑calibrator factors, decay correction policies.

**Fields:** - id **MUST** — urn:tides:calibration:<UUID> - scannerFactor\_Bq\_per\_count **MUST** - decayCorrectionPolicy **MUST** — acq-start | frame-mid | injection-start | none - crossCal **SHOULD** — { doseCalibratorId, factor, date }

### 2.2.5 TidesPKModel

PK fit configuration and results (voxelwise or ROI‑wise).

**Fields:** - id **MUST** — urn:tides:pkmodel[:<flavor>]:<UUID> where <flavor> ∈ monoexp | biexp | twoComp | spline - scope **MUST** — voxel | roi - priors **MAY** — documented priors - fit **MUST** — method, weights, parameter estimates, covariance (for delta‑method) - tia **SHOULD** — per voxel/ROI results or a reference to a map

**Constraints:** Sampling adequacy WARNs (§1.3) apply; covariance **MUST** reflect actual weights.

### 2.2.6 TidesDoseMap

Voxel absorbed dose (primary artifact for Profiles A/B).

**Fields:** - id **MUST** — urn:tides:dosemap[:<nuclide>]:<UUID> - rtDose **MUST** — reference to DICOM RTDOSE or embedded grid pointer - frameOfReferenceUID **MUST** - kernel **MUST** — { formalism: S-value|MC, nuclide, medium, grid, version } - source **MUST** — TidesPKModel URN - flashCoverage\_pc **SHOULD** — global coverage; per‑ROI coverage **SHOULD**

**Constraints:** Lossless inputs only; resampling **MUST** retain transform provenance.

### 2.2.7 TidesDoseReport

Organ/ROI dose summaries, safety checks, uncertainties, and references back to TidesDoseMap.

**Fields:** - id **MUST** — urn:tides:dosereport:<UUID> - series[] **MUST** — items with { doseMap, organSummaries[] } - organSummaries[] **MUST** — items with { roiCode, roiCodeText, Dmean\_Gy, Dmax\_Gy, Dx (SHOULD), Vx (SHOULD), uncertainty\_pc (MUST), uncertaintyMethod (MUST) } - policyResults **SHOULD** — { oar: passes|warn|fails, flash: coverage }

**Constraints:** Units **MUST** be UCUM; see Chapter 12 for reporting templates.

### 2.2.8 TidesOutcome

Longitudinal outcomes (response, toxicity) tied to the same subject and study.

**Fields:** - id **MUST** — urn:tides:outcome:<UUID> - timepoint **MUST** — ISO‑8601 date/time - response **SHOULD** — iRECIST/PERCIST‑ext terms - toxicity[] **SHOULD** — CTCAE coded items - links **MAY** — imaging or labs underlying the assessment

## 2.3 Cardinalities & Constraints (Normative Table)

| Parent → Child | Cardinality | Notes |
| --- | --- | --- |
| TidesStudy → TidesImaging | 1..N | At least one imaging cohort per study |
| TidesImaging → TidesSeg | 0..N | Optional; Profiles A/B require coded ROIs for OARs |
| TidesImaging → TidesCalibration | 0..1 | Recommended; WARN if absent |
| TidesStudy → TidesPKModel | 1..N | At least one model per nuclide/ROI scope |
| TidesPKModel → TidesDoseMap | 1..N | Different kernels/media/registrations |
| TidesDoseMap → TidesDoseReport | 1..N | Multiple reporting policies allowed |
| TidesStudy → TidesOutcome | 0..N | Optional but recommended longitudinally |

**Validator rules:** cardinality-min (ERROR), cardinality-max (ERROR if >N where N fixed), link-exists (ERROR), link-type (ERROR on wrong type).

## 2.4 Minimal JSON Serializations (Canonical)

### 2.4.1 Bundle (Profile A minimal)

{  
 "specVersion": "1.0.0",  
 "id": "urn:tides:bundle:7f2...",  
 "study": "urn:tides:study:1b2f9c60-0e8a-4d6d-9e39-7c4e4d2f9a6a",  
 "resources": [  
 {"resourceType": "TidesStudy", "id": "urn:tides:study:1b2f...", "subject": "urn:tides:subject:9a2f..."},  
 {"resourceType": "TidesImaging", "id": "urn:tides:imaging:aa11...", "frameOfReferenceUID": "1.2.3", "timings": {"injectionStart": "2025-09-25T11:03:27Z", "frames": [{"tMid\_s":600,"duration\_s":120}]}},  
 {"resourceType": "TidesSeg", "id": "urn:tides:seg:bb22...", "roi": [{"code":"74281007","codeSystem":"SNOMED","text":"liver"}]},  
 {"resourceType": "TidesPKModel", "id": "urn:tides:pkmodel:monoexp:cc33...", "scope":"roi", "fit": {"method":"WLS","params":{"A0":1.2e6,"lambda":3.1e-5}, "covariance":[[...]]}},  
 {"resourceType": "TidesDoseMap", "id": "urn:tides:dosemap:177Lu:dd44...", "rtDose":"dicom:1.2.840...", "kernel": {"formalism":"S-value","nuclide":"177Lu","grid":"2mm","version":"v1"}, "source":"urn:tides:pkmodel:monoexp:cc33..."},  
 {"resourceType": "TidesDoseReport", "id": "urn:tides:dosereport:ee55...", "series":[{"doseMap":"urn:tides:dosemap:177Lu:dd44...","organSummaries":[{"roiCode":"74281007","roiCodeText":"liver","Dmean\_Gy":18.2,"Dmax\_Gy":21.5,"uncertainty\_pc":9.1,"uncertaintyMethod":"delta-method"}]}]}  
 ],  
 "provenance": {"software":"tides-cli","version":"1.0.0","hash":"<sha>","inputs":["dicom:..."],"policyPacks":[{"name":"policy-oar","version":"1.0.0","specMajor":1}]}  
}

## 2.5 Ontology Bindings (ROIs, Outcomes, Units)

### 2.5.1 ROIs

* **Codesystems:** SNOMED CT (preferred), RadLex (acceptable).
* **Normative examples:** 74281007|Liver, 64033007|Kidney structure.
* **Free text:** roiCodeText is mandatory for human readability but does not substitute the code.

### 2.5.2 Outcomes

* **Toxicity:** CTCAE v5 preferred. Store { code, grade, onsetDate } per item.
* **Response:** iRECIST or PERCIST-ext label + measurement references.

### 2.5.3 Units

* **UCUM only** per §1.1; validator enforces tokens.

**Validator rules:** codesystem-roi (WARN/ERROR per profile), codesystem-outcome (WARN), unit-ucum (ERROR).

## 2.6 Interoperability Mappings (DICOM ↔ FHIR ↔ OMOP)

### 2.6.1 DICOM Essentials (normative pointers)

* **FoR**: (0020,0052) **MUST** be present and consistent for RTDOSE and referenced images.
* **SEG**: SegmentedPropertyCategoryCodeSequence (0062,0003) and SegmentedPropertyTypeCodeSequence (0062,000F) **MUST** carry codes that map to roi.code.
* **RTDOSE**: DoseGridScaling (3004,000E), GridFrameOffsetVector (3004,000C) **MUST** be populated; link to parent via ReferencedFrameOfReferenceSequence (3006,0010) and ReferencedImageSequence.

### 2.6.2 FHIR Profiles (minimal working set)

| TIDeS Resource | FHIR Resource | Key Elements |
| --- | --- | --- |
| TidesImaging | ImagingStudy | identifier with SeriesInstanceUIDs; endpoint to VNA |
| TidesSeg | Observation (or ImagingSelection for references) | code from SNOMED/RadLex; derivedFrom ImagingStudy |
| TidesDoseReport | DiagnosticReport | result → Observation(Absorbed dose LOINC 89469‑4); conclusionCode policy flags |
| Dose per ROI | Observation | valueQuantity in Gy; bodySite ROI code |
| Provenance | Provenance | agent (software), entity (inputs) |

**Example (Absorbed Dose Observation):**

{  
 "resourceType": "Observation",  
 "meta": {"profile": ["http://tides.org/fhir/StructureDefinition/AbsorbedDose"]},  
 "code": {"coding": [{"system":"http://loinc.org","code":"89469-4","display":"Absorbed dose"}]},  
 "valueQuantity": {"value": 8.3, "system":"http://unitsofmeasure.org", "code":"Gy"},  
 "bodySite": {"coding":[{"system":"http://snomed.info/sct","code":"74281007","display":"Liver"}]}  
}

### 2.6.3 OMOP Extension (DDL)

CREATE TABLE tides\_study (  
 tides\_study\_id SERIAL PRIMARY KEY,  
 person\_id INT NOT NULL,  
 study\_uid TEXT NOT NULL UNIQUE,  
 nuclide TEXT, agent TEXT, protocol\_id TEXT, start\_ts TIMESTAMP, end\_ts TIMESTAMP  
);  
  
CREATE TABLE tides\_roi\_dose (  
 tides\_roi\_dose\_id SERIAL PRIMARY KEY,  
 study\_uid TEXT NOT NULL REFERENCES tides\_study(study\_uid),  
 roi\_code TEXT NOT NULL, roi\_text TEXT,  
 dmean\_gy NUMERIC, dmax\_gy NUMERIC,  
 v10gy NUMERIC, dx\_gy NUMERIC,  
 uncertainty\_pc NUMERIC, uncertainty\_method TEXT,  
 dosemap\_uri TEXT, pk\_model\_id TEXT,  
 created\_at TIMESTAMP DEFAULT now()  
);  
  
CREATE TABLE tides\_provenance (  
 tides\_prov\_id SERIAL PRIMARY KEY,  
 study\_uid TEXT NOT NULL,  
 software TEXT, version TEXT, hash TEXT,  
 inputs JSONB, params JSONB, policy\_packs JSONB  
);

**Constraints:** study\_uid **MUST** be a TIDeS URN stored verbatim.

## 2.7 Immutability, Versioning & Lineage

* Artifacts are **immutable** once published. Corrections produce a **new** id and set provenance.parent to the prior id.
* versionTag **MAY** annotate minor editorial fixes that do **not** change numeric outputs; otherwise use a new id.
* The validator **MUST** verify that no resource self‑references or forms a cycle (graph-dag).

**Lineage example:**

id: urn:tides:dosereport:ee55...  
provenance:  
 parent: urn:tides:dosereport:9c1a... # previous run  
 software: tides-cli  
 version: 1.0.1  
 params: { model: monoexp, kernel: S-value-177Lu-v1 }

## 2.8 Attachments & Binary Payloads

* Binary DICOM/RTDOSE/SEG **SHOULD** live in a VNA/object store. TIDeS resources **SHOULD** carry attachments[] items with { uri, type, description, hash }.
* Hashes **MUST** be SHA‑256 of the exact binary served.
* attachments **MUST NOT** contain PHI if the bundle is intended for public sharing.

**Validator rules:** attachment-hash (WARN if missing), attachment-type (WARN if unknown).

## 2.9 Error Taxonomy & Validation Outcomes (Model Layer)

* **ERROR**: Violates identifiers, missing FoR for spatial items, dangling links, non‑UCUM units in required fields, wrong cardinality.
* **WARN**: Sampling adequacy, missing recommended policy packs, missing uncertainty breakdown (but not the total), unknown outcome code systems.
* **INFO**: Deprecated fields in grace window, software version drifts.

## 2.10 Must‑Support Matrix (Profiles A/B/C)

| Capability | A (Clinical‑Full) | B (Research‑Voxel) | C (Legacy‑Organ) |
| --- | --- | --- | --- |
| id URNs | **MUST** | **MUST** | **MUST** |
| specVersion | **MUST** | **MUST** | **MUST** |
| TidesImaging.frameOfReferenceUID | **MUST** | **MUST** | **MAY** |
| TidesSeg.roi.codeSystem | **MUST** (SNOMED/RadLex) | **SHOULD** | **MAY** |
| TidesDoseMap.rtDose | **MUST** | **MUST** | **N/A** |
| TidesDoseReport.organSummaries[].uncertainty\_pc | **MUST** | **MUST** | **MUST** |
| policyPacks | **MUST** (OAR) | **SHOULD** | **SHOULD** |
| FHIR DiagnosticReport export | **MUST** | **SHOULD** | **MAY** |
| OMOP rows | **SHOULD** | **SHOULD** | **MAY** |

**Validator rules:** profile‑scoped rule lists are evaluated before PASS badges get issued.

## 2.11 Worked Example (End‑to‑End)

**Narrative:** A patient undergoes 177Lu‑DOTATATE therapy. Dynamic SPECT at 10 min, 60 min, 6 h. ROIs for liver and kidneys segmented. Mono‑exponential PK per ROI. Voxel dose computed via S‑values. Dose report generated with uncertainty and OAR policy evaluation.

**Bundle excerpt (YAML):**

specVersion: 1.0.0  
id: urn:tides:bundle:fc1f...  
study: urn:tides:study:1b2f...  
resources:  
 - resourceType: TidesStudy  
 id: urn:tides:study:1b2f...  
 subject: urn:tides:subject:9a2f...  
 context: { indication: NET, nuclide: 177Lu, agent: DOTATATE }  
 - resourceType: TidesImaging  
 id: urn:tides:imaging:aa11...  
 frameOfReferenceUID: 1.2.826...  
 timings:  
 injectionStart: 2025-09-25T11:03:27Z  
 frames: [{tMid\_s:600,duration\_s:120},{tMid\_s:3600,duration\_s:180},{tMid\_s:21600,duration\_s:240}]  
 - resourceType: TidesSeg  
 id: urn:tides:seg:bb22...  
 roi:  
 - { code: "74281007", codeSystem: "SNOMED", text: "liver" }  
 - { code: "64033007", codeSystem: "SNOMED", text: "kidney" }  
 - resourceType: TidesPKModel  
 id: urn:tides:pkmodel:monoexp:cc33...  
 scope: roi  
 fit:  
 method: WLS  
 params: { A0: 1.4e7, lambda: 2.9e-5 }  
 covariance: [[1.2e4, -3.1e-2], [-3.1e-2, 5.4e-8]]  
 - resourceType: TidesDoseMap  
 id: urn:tides:dosemap:177Lu:dd44...  
 rtDose: dicom:1.2.840.10008.5.1.4.1.1.481.2:7.5.9  
 frameOfReferenceUID: 1.2.826...  
 kernel: { formalism: S-value, nuclide: 177Lu, medium: ICRP-soft, grid: 2mm, version: v1 }  
 source: urn:tides:pkmodel:monoexp:cc33...  
 flashCoverage\_pc: 3.1  
 - resourceType: TidesDoseReport  
 id: urn:tides:dosereport:ee55...  
 series:  
 - doseMap: urn:tides:dosemap:177Lu:dd44...  
 organSummaries:  
 - { roiCode: "74281007", roiCodeText: liver, Dmean\_Gy: 18.2, Dmax\_Gy: 21.5, Vx: {"10Gy": 0.21}, uncertainty\_pc: 9.1, uncertaintyMethod: delta-method }  
 - { roiCode: "64033007", roiCodeText: kidney, Dmean\_Gy: 12.8, Dmax\_Gy: 16.4, uncertainty\_pc: 10.3, uncertaintyMethod: bootstrap }  
 policyResults: { oar: passes }  
provenance:  
 software: tides-cli  
 version: 1.0.0  
 hash: 5f6e...  
 inputs: ["dicom:..."]  
 params: { model: monoexp, kernel: S-value-177Lu-v1 }  
 policyPacks: [ { name: policy-oar, version: 1.0.0, specMajor: 1 } ]

Running the validator yields A-PASS (all MUST satisfied, one INFO on missing OMOP export in this example).

## 2.12 Traceability Matrix (Extract)

2.1,id-urn-format,/id,case\_pass\_minimal  
2.2,link-exists,/resources[\*]/source,case\_registration\_link\_missing  
2.3,cardinality-min,/resources[type=TidesPKModel],case\_sampling\_inadequate  
2.6,codesystem-roi,/resources[type=TidesSeg]/roi[\*]/codeSystem,case\_registration\_ok  
2.10,profile-matrix,/resources,case\_legacy\_profile\_C

## 2.13 FAQs (Operational)

**Q: Can I use URLs instead of URNs?**  
A: Use URNs for identity and **add** URLs in attachments[]. Identity must not rely on a live endpoint.

**Q: How do I represent multiple nuclides in a combination therapy?**  
A: Separate TidesPKModel and TidesDoseMap per nuclide; aggregate in a higher‑level TidesDoseReport series array with nuclide annotated in kernel.

**Q: Our ROI system is hospital‑local.**  
A: Map local codes to SNOMED/RadLex and include local code under extensions.localCode (non‑normative). The validator will WARN if canonical code missing under Profile A.

## 2.14 Chapter Summary

* The **DAG** anchors theranostics artifacts with durable **URNs**.
* Each node has clear cardinalities and profile expectations.
* Mappings to **DICOM**, **FHIR**, and **OMOP** are concrete and testable.
* Immutability + provenance ensure scientific integrity and regulatory defensibility.

### Appendix 2‑A: JSON Schema Skeletons (Pointers)

Full schemas live in /schemas; below are normative skeletons to guide implementers.

**tides-study.schema.json (excerpt)**

{  
 "$id": "https://tides.org/schemas/1.0.0/tides-study.schema.json",  
 "$schema": "https://json-schema.org/draft/2020-12/schema",  
 "title": "TidesStudy",  
 "type": "object",  
 "required": ["id","specVersion","subject"],  
 "properties": {  
 "id": {"type":"string","pattern":"^urn:tides:study:[0-9a-fA-F-]{36}$"},  
 "specVersion": {"type":"string","const":"1.0.0"},  
 "subject": {"type":"string","pattern":"^urn:tides:subject:[0-9a-fA-F-]{36}$"}  
 }  
}

**tides-dose-report.schema.json (expanded excerpt)**

{  
 "$id": "https://tides.org/schemas/1.0.0/tides-dose-report.schema.json",  
 "$schema": "https://json-schema.org/draft/2020-12/schema",  
 "title": "TIDeS Dose Report",  
 "type": "object",  
 "required": ["id","specVersion","subject","series"],  
 "properties": {  
 "series": {  
 "type": "array",  
 "items": {  
 "type": "object",  
 "required": ["doseMap","organSummaries"],  
 "properties": {  
 "doseMap": {"type":"string","pattern":"^urn:tides:dosemap(:[A-Za-z0-9]+)?:[0-9a-fA-F-]{36}$"},  
 "organSummaries": {  
 "type":"array",  
 "items":{  
 "type":"object",  
 "required":["roiCode","Dmean\_Gy","uncertainty\_pc"],  
 "properties": {  
 "roiCode": {"type":"string"},  
 "Dmean\_Gy": {"type":"number"},  
 "uncertainty\_pc": {"type":"number"}  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
}

### Appendix 2‑B: ASCII ER Diagram

[Study] 1---N [Imaging] 1---N [Seg]  
 |   
 |\   
 | \\_\_N [PKModel] 1---N [DoseMap] 1---N [DoseReport]  
 |   
 \\_\_N [Outcome]

### Appendix 2‑C: Validator Rule IDs (Model Layer)

* id-namespace, id-uuid, id-uniqueness, id-referential
* graph-dag, cardinality-min, cardinality-max, link-type
* codesystem-roi, codesystem-outcome
* attachment-hash, attachment-type

**End of Chapter 2 (Normative).**

# TIDeS Handbook — Chapter 3

## Imaging & Spatial Semantics (Normative Chapter)

**Purpose.** Define exactly *how* images, segmentations, registrations, parametric maps, and voxel‑dose grids are represented, linked, and trusted in TIDeS. This chapter makes geometry **unambiguous**, timing **auditable**, and resampling **explainable** across vendors and sites.

**Audience.** Nuclear medicine & PET/SPECT physicists, medical imaging engineers, radiation oncology/RT‑DICOM experts, PACS/VNA architects, research pipeline developers, regulators.

**Outcome.** By the end, you can (a) export, validate, and persist *lossless* DICOM inputs; (b) tie all spatial objects to a common **FrameOfReferenceUID** (FoR); (c) store transforms instead of “hoping” registrations match; (d) deliver voxel dose in **RTDOSE** correctly aligned; (e) publish coded DICOM **SEG** labelmaps; (f) ship parametric maps (SUV, Ki, TIA) with **Real‑World Value Mapping**; and (g) pass every Profile A/B/C imaging rule in the validator.

**Normative keywords:** **MUST**, **SHOULD**, **MAY**.

## 3.0 Scope & Principles (Normative)

1. **Lossless First.** All dosimetry inputs (PET/SPECT, CT, MR, SEG, registrations, RTDOSE) **MUST** be stored and exchanged in **lossless** encodings. Use explicit VR little‑endian or lossless image compression. Lossy encodings for quantitation inputs are **prohibited**.
2. **One Truth of Space.** Every spatial object **MUST** carry and respect a DICOM **FrameOfReferenceUID (FoR)**. Any change of grid **MUST** reference an explicit **transform** object (rigid/affine/deformable).
3. **Explicit Corrections.** Acquisition & reconstruction corrections (decay, attenuation, scatter, normalization, dead‑time) **MUST** be declared in headers and preserved through export. Silence is non‑compliance.
4. **Value Mapping is Law.** Numeric values in parametric objects **MUST** be accompanied by **Real‑World Value Mapping** or equivalent embedded units/scales.
5. **Coded ROIs.** Anatomical/functional ROIs **MUST** be coded (SNOMED CT or RadLex) in DICOM SEG and mirrored in TIDeS resources.
6. **Registration is Data.** Registrations are artifacts—not assumptions. Store the registration IOD (or transform grid), reference it by UID, and capture provenance.

Validator hooks: lossless-required, registration-fuid, registration-transform-provenance, seg-coded-meanings, parametric-rwvm, rtdose-for-match, dicom-corrections-declared.

## 3.1 DICOM Imaging Objects (Authoritative)

### 3.1.1 PET/SPECT/NM Series

* **MUST** include radionuclide & administered activity metadata and **decay correction policy** (e.g., to acquisition start or frame mid).
* **MUST** indicate applied corrections (attenuation, scatter, randoms, normalization, dead‑time) via the standard *Corrected Image* flags.
* **SHOULD** provide dynamic framing details sufficient to reconstruct {tMid\_s, duration\_s} (see Chapter 1/8).
* **MUST** use lossless transfer syntaxes for voxel data used in quantitation.

### 3.1.2 CT / MR for Attenuation & Anatomy

* **CT** used for AC or density mapping **MUST** be exported lossless with correct **HU calibration** and body kernel documentation.
* **MR** attenuation surrogates (if used) **MUST** declare the vendor’s μ‑map generation method; any atlas or ZTE specifics **SHOULD** be documented in provenance.

### 3.1.3 Dynamic / Gated Data

* If respiratory or cardiac gating is used, the gating scheme (phase vs amplitude) and reference frame **MUST** be recorded.
* Dynamic frames **MUST** be reconstructable to a flat list of intervals with absolute or relative times mapped to injectionStart.

**Validator rules:** dicom-corrections-declared (ERROR if unknown), lossless-required (ERROR for inputs), timing-frames (ERROR if irrecoverable).

## 3.2 Coordinate System & Geometry

### 3.2.1 Frame of Reference (FoR)

* Every spatial object (images, SEG, RTDOSE, parametric maps) **MUST** carry the DICOM **FrameOfReferenceUID**.
* Derived objects **MUST** specify the **source FoR** and the transform used when the FoR differs.

### 3.2.2 Physical Coordinates

* Physical coordinates are derived from *Image Position (Patient)*, *Image Orientation (Patient)*, and *Pixel Spacing*/spacing between slices. Implementations **MUST** compute and validate the patient‑based coordinates before resampling.
* **Tolerance:** For aligned series expected to share a grid, voxel center differences **MUST NOT** exceed **0.25×** the smallest voxel dimension.

### 3.2.3 Grid Requirements

* **RTDOSE** grids **MUST** declare spacing and grid offsets;
* Parametric maps **MUST** declare pixel representation (float recommended), slope/intercept or RWVM, and units;
* SEG objects **MUST** align to a referenced image grid or declare the transform to the dose grid.

**Validator rules:** registration-fuid (ERROR), grid-tolerance (WARN/ERROR per profile), parametric-rwvm (ERROR).

## 3.3 Registration: Rigid, Affine, Deformable

### 3.3.1 Objects & References

* **Rigid/Affine** registrations **SHOULD** use the DICOM Spatial Registration IOD and be referenced by SOP Instance UID.
* **Deformable** registrations **MUST** be stored as Deformable Spatial Registration objects or equivalent vector fields with declared direction, units, spacing, and origin.
* All transformations **MUST** include provenance { software, version, parameters, hash } and **MUST** be immutable.

### 3.3.2 Semantics & Directionality

* Define transforms as *from source FoR → target FoR*. The direction **MUST** be explicit. If an inverse is used, either store it or declare the inversion method and numeric tolerance.

### 3.3.3 Interpolation and Resampling

* **SHOULD** use tri‑linear or higher‑order (B‑spline) interpolation for scalar images and **NEAREST** for labelmaps.
* Resampling **MUST** record: interpolation, padding strategy, extrapolation behavior, and voxel alignment policy (centered vs edge‑aligned).

### 3.3.4 Quality & Acceptance Metrics

* Report at least one **overlap metric** (e.g., Dice) for key ROIs and one **geometric error** (e.g., TRE or Hausdorff distance).
* **Profile A**: Dice ≥ **0.90** for rigidly related anatomy (e.g., liver, kidneys) is **RECOMMENDED**; deviations **MUST** be justified.

**Validator rules:** registration-transform-provenance (ERROR), registration-quality (WARN with thresholds), label-resample-method (WARN if not NEAREST).

## 3.4 Segmentation (DICOM SEG)

### 3.4.1 Coding & Semantics (Normative)

* Each segment **MUST** declare:
  + SegmentNumber, SegmentLabel,
  + **Coded Meanings** via *Segmented Property Type* and *Category* (SNOMED CT/RadLex codes),
  + SegmentAlgorithmType and SegmentAlgorithmName.
* The SEG **MUST** reference the source image frames and **FoR**.

### 3.4.2 Binary vs Fractional

* **Binary** masks are normative. **Fractional** segmentations **MAY** be used for partial‑volume probability maps but **MUST** declare fractional type (e.g., *probability*) and range.

### 3.4.3 Label Hygiene

* Avoid overlapping anatomy unless clinically intentional; if overlaps exist, declare a **label policy** (exclusive | inclusive | hierarchical) in metadata.
* Provide **minimum size filters** for speckle suppression and document morphology operations (erode/dilate) where applied.

### 3.4.4 Surfaces & RTSTRUCT

* Surface meshes **MAY** accompany SEG for visualization; ensure the mesh was generated from the same SEG and FoR, and record the pipeline.
* RTSTRUCT legacy contours **MAY** be ingested for Profile C; they **SHOULD** be converted to SEG with codes.

**Validator rules:** seg-coded-meanings (ERROR A/B), seg-fractional-declare (WARN), seg-source-ref (ERROR), seg-overlap-policy (WARN).

## 3.5 Voxel Dose (DICOM RTDOSE)

### 3.5.1 Storage & Alignment

* Voxel dose **MUST** be stored or referenced via **RTDOSE** and aligned to a declared FoR.
* **MUST** include grid offsets (frame offsets), dose scaling, units (Gy), and a pointer to the source planning/referenced images.

### 3.5.2 Dose Type & Summation

* **Dose Units:** Gy only.
* **Dose Type:** PHYSICAL (normative). If alternative types are stored, they **MUST** be clearly labeled and **MUST NOT** be used for absorbed dose reporting.
* **Summation:** If composing dose from multiple maps (e.g., multiple administrations), the summation rule **MUST** be declared.

### 3.5.3 Grid Policy

* **Preferred grid:** isotropic ≤ **3 mm** for voxel‑level dosimetry; if coarser, justify and document kernel resolution.
* **Tolerances:** Dose grid alignment relative to reference anatomy grid **MUST** be within **0.25×** voxel size (center‑to‑center).

**Validator rules:** rtdose-for-match (ERROR), unit-dose-gy (ERROR), dose-grid-tolerance (WARN/ERROR), dose-type-physical (ERROR).

## 3.6 Parametric Maps & Real‑World Value Mapping (RWVM)

### 3.6.1 What Counts as a Parametric Map

* **SUVbw**, **SUVlbm**, **SUVbsa**, **Ki** (Patlak), **TIA** (time‑integrated activity), perfusion or kinetic coefficients.
* These objects **MUST** store *floating‑point* pixel data or an integer + slope/intercept pair and **MUST** embed **RWVM** with a **UCUM** unit.

### 3.6.2 RWVM Requirements (Normative)

* Each parametric map **MUST** define:
  + mapping slope/intercept **or** an explicit real‑world LUT,
  + the **measurement unit code** (UCUM),
  + value range expectations.
* If multiple ranges exist (e.g., masked background), use separate RWVM items.

### 3.6.3 SUV Semantics

* **SUVbw** requires body weight at acquisition; **SHOULD** include the exact formula used.
* **SUVlbm/SUVbsa** **MUST** declare the calculation model (e.g., Janmahasatian for LBM).

**Validator rules:** parametric-rwvm (ERROR), unit-ucum (ERROR), suv-formula-declared (WARN).

## 3.7 Compression, Bit‑Depth & Fidelity

* **Inputs for quantitation** (PET/SPECT/CT used for AC) **MUST** be lossless. Acceptable examples include Explicit VR Little‑Endian and widely used lossless image compressions.
* **Bit‑depth:** Keep native precision (e.g., 16‑bit CT); avoid unnecessary scaling that reduces dynamic range.
* **RTDOSE/Parametric:** Use floating point where supported; otherwise declare slope/intercept precisely.

**Validator rules:** lossless-required (ERROR), bitdepth-downgrade (WARN).

## 3.8 Acquisition & Reconstruction Corrections (Declaring the Truth)

* **Decay Correction:** declare target (injection start, frame mid, acquisition start) and half‑life used.
* **Attenuation:** declare CT‑based μ‑map vs MR surrogate (ZTE/atlas), energy scaling policy.
* **Scatter/Randoms/Normalization/Dead‑time:** indicate applied corrections.
* **Motion:** record gating and motion correction (if applied) as part of provenance.

**Validator rules:** dicom-corrections-declared (ERROR), half-life-mismatch (WARN if nuclide/half‑life inconsistent), ac-policy-coherence (WARN if AC on without CT/MR μ‑map declaration).

## 3.9 HU→Density & Material Maps (Optional but Controlled)

* If HU→ρ conversion is used for kernel media selection or dose heterogeneity corrections, the mapping curve **MUST** be recorded (points or formula) and the source (e.g., ICRU/ICRP references) documented.
* If not used, default to soft‑tissue assumptions and declare this explicitly.

**Validator rules:** hu-density-declared (WARN), kernel-media-consistency (WARN).

## 3.10 End‑to‑End Spatial Pipeline (Reference SOP)

1. **Ingest** PET/SPECT + CT/MR series (lossless); verify Corrected Image flags.
2. **Normalize clocks** to injectionStart; reconstruct frames → {tMid\_s, duration\_s}.
3. **Check FoR** consistency; if multiple FoRs, plan registrations.
4. **Segment** OARs/targets → DICOM SEG with codes; record algorithm.
5. **Register** source→dose grid (rigid/affine; deformable if justified); store transform & metrics.
6. **Resample** activity/PK inputs only when necessary; record interpolation & padding.
7. **Compute** parametric maps with RWVM; persist units.
8. **Generate** RTDOSE grid in target FoR; confirm grid tolerances.
9. **Link** SEG→RTDOSE→DoseReport; compute ROI stats.
10. **Validate** bundle; remediate errors; publish with provenance and policy packs.

## 3.11 Quality Assurance & Test Artifacts

### 3.11.1 Phantoms & Cross‑Cal

* **NEMA/IEC body phantoms** for PET/SPECT **SHOULD** be scanned periodically;
* **Cross‑calibration** to dose calibrator activity **MUST** be current and recorded (see TidesCalibration).

### 3.11.2 Spatial & Registration QA

* Use synthetic shifted/rotated datasets with known ground truth to assert transform accuracy.
* Track Dice/TRE across releases; regression test in CI.

### 3.11.3 Header Consistency

* CI step to assert FoR, spacing, orientation, and corrections for all example datasets.

**Validator rules:** calibration-present (WARN), qa-metrics-present (WARN).

## 3.12 Canonical Examples (Copy‑Paste)

### 3.12.1 Geometry Block (JSON)

{  
 "frameOfReferenceUID": "1.2.826.0.1.3680043.2.1125.1.1234.5678",  
 "grid": { "spacing\_mm": [2.0, 2.0, 2.0], "size": [256, 256, 160] },  
 "registration": {  
 "type": "deformable",  
 "ref": "dicom:deformable-reg:2.16.840....",  
 "provenance": { "software": "tides-reg", "version": "1.0.0", "hash": "<sha>" },  
 "interpolation": { "scalar": "trilinear", "labels": "nearest" }  
 }  
}

### 3.12.2 SEG Header (YAML excerpt)

seg:  
 id: dicom:1.2.840.10008.5.1.4.1.1.66.4:... # Segmentation Storage SOP  
 frameOfReferenceUID: 1.2.826...  
 segments:  
 - number: 1  
 label: liver  
 code: { system: SNOMED, value: 74281007, display: Liver }  
 algorithm: { type: semi-automatic, name: live-wire }

### 3.12.3 Parametric Map RWVM (JSON excerpt)

{  
 "map": "SUVbw",  
 "pixelRepresentation": "float32",  
 "realWorldValue": {  
 "unit": { "system": "http://unitsofmeasure.org", "code": "{SUVbw}" },  
 "slope": 1.0,  
 "intercept": 0.0  
 }  
}

### 3.12.4 RTDOSE Linkage (YAML excerpt)

rtdose:  
 sopClass: RTDOSE  
 frameOfReferenceUID: 1.2.826...  
 doseUnits: Gy  
 grid: { spacing\_mm: [2.0,2.0,2.0], size: [256,256,160], offsets\_mm: [0.0,0.0,0.0] }  
 references:  
 images: [ dicom:pet:..., dicom:ct:... ]  
 registration: dicom:reg:...

## 3.13 Validator Rule Set (Imaging Layer)

* lossless-required — Inputs for quantitation must be lossless (ERROR).
* dicom-corrections-declared — Corrections/decay policy declared (ERROR).
* registration-fuid — Every spatial artifact must have FoR (ERROR).
* registration-transform-provenance — Transform stored & referenced (ERROR).
* grid-tolerance — Within 0.25× voxel size (WARN/ERROR).
* rtdose-for-match — RTDOSE must align to FoR (ERROR).
* parametric-rwvm — Parametric maps must have RWVM/units (ERROR).
* seg-coded-meanings — SEG segments must be coded (ERROR for A/B, WARN C).
* label-resample-method — Label resampling must be NEAREST (WARN).
* dose-type-physical — RTDOSE reports physical dose in Gy (ERROR).
* hu-density-declared — HU→ρ mapping declared if applied (WARN).

## 3.14 Must‑Support Matrix (Profiles A/B/C)

| Capability | A (Clinical‑Full) | B (Research‑Voxel) | C (Legacy‑Organ) |
| --- | --- | --- | --- |
| Lossless inputs | **MUST** | **MUST** | **SHOULD** |
| FoR on all objects | **MUST** | **MUST** | **MAY** |
| Stored registrations | **MUST** | **MUST** | **SHOULD** |
| SEG coded ROIs | **MUST** | **SHOULD** | **MAY** |
| Parametric RWVM | **MUST** | **MUST** | **SHOULD** |
| RTDOSE alignment | **MUST** | **MUST** | **N/A** |
| QA metrics | **SHOULD** | **SHOULD** | **MAY** |

## 3.15 TIDeS‑CHK‑3 — Imaging Readiness Checklist

* All imaging inputs are lossless; bit‑depth preserved.
* FoR present on PET/SPECT/CT/MR, SEG, parametric, and RTDOSE.
* Registrations stored (rigid/affine/deformable) with provenance and direction.
* SEG segments coded (SNOMED/RadLex) and aligned to source image.
* Parametric maps carry RWVM and UCUM units.
* RTDOSE grid spacing/offsets declared; aligns within tolerance to anatomy grid.
* Corrections and decay policy declared; half‑life matches nuclide.
* HU→ρ curve declared if used; otherwise default annotated.
* QA phantom/cross‑cal logs linked in provenance.

## 3.16 Traceability (Extract)

3.0,lossless-required,/interops/dicom,case\_pass\_minimal  
3.2,registration-fuid,/resources[type=TidesDoseMap]/frameOfReferenceUID,case\_registration\_link\_missing  
3.3,registration-transform-provenance,/resources[type=TidesDoseMap]/registration,case\_registration\_ok  
3.4,seg-coded-meanings,/resources[type=TidesSeg]/roi[\*]/codeSystem,case\_legacy\_profile\_C  
3.5,rtdose-for-match,/resources[type=TidesDoseMap]/rtDose,case\_sampling\_inadequate  
3.6,parametric-rwvm,/resources[type=ParametricMap]/realWorldValue,case\_ucum\_wrong\_unit\_text

## 3.17 FAQs (Operational)

**Q: Can we regrid CT to PET to simplify processing?**  
A: Yes, but store the registration and record interpolation methods. Do not overwrite original CT; keep provenance for both.

**Q: Are RTSTRUCTs acceptable?**  
A: For Profile C legacy, yes; convert to SEG with codes for A/B.

**Q: Can we apply lossy compression for archiving?**  
A: Not for quantitation inputs. For secondary *visualization* copies you may, but they are out of scope of TIDeS conformance.

**Q: How to handle mixed FoRs from multi‑day imaging?**  
A: Keep each day’s FoR and register to a chosen dose FoR; document accumulated transforms.

## 3.18 Chapter Summary

* Spatial truth in TIDeS is **declared, not assumed**: FoR everywhere, transforms stored, units mapped, grids aligned, corrections explicit.
* With these rules, voxelwise dosimetry becomes portable, auditable, and reproducible across centers and vendors.

### Appendix 3‑A: Numeric Tolerances (Suggested)

* Grid center alignment: ≤ 0.25× min voxel size (WARN at 0.25×, ERROR at ≥ 0.5×).
* Registration quality: Dice ≥ 0.90 for rigid anatomy (WARN < 0.90), TRE ≤ 2–3 mm depending on region.
* Dose grid resolution: ≤ 3 mm isotropic preferred for abdominal OARs.

### Appendix 3‑B: Interpolation Policy Table

| Data Type | Recommended Interp | Notes |
| --- | --- | --- |
| Scalar images (PET/SPECT/CT/MR) | Tri‑linear or B‑spline | Avoid ringing in high‑contrast edges |
| Labelmaps (SEG) | Nearest neighbor | Preserve label identity |
| Parametric maps | Tri‑linear/B‑spline | Preserve RWVM semantics |
| Vector fields | Linear | Ensure correct component spacing/units |

### Appendix 3‑C: Registration Provenance Template (YAML)

registration:  
 id: urn:tides:reg:...  
 type: deformable  
 direction: source->target  
 sourceFoR: 1.2.826...  
 targetFoR: 1.2.826...  
 metrics: { dice\_liver: 0.93, tre\_kidney\_mm: 2.1 }  
 software: { name: tides-reg, version: 1.0.0 }  
 parameters: { grid: 5mm, bspline\_order: 3, iterations: 200 }  
 hash: <sha256>

### Appendix 3‑D: Example RWVM for Ki (Patlak) (JSON)

{  
 "map": "Ki",  
 "pixelRepresentation": "float32",  
 "realWorldValue": {  
 "unit": { "system": "http://unitsofmeasure.org", "code": "1/min" },  
 "slope": 1.0,  
 "intercept": 0.0  
 },  
 "fit": { "method": "Patlak", "framesUsed": [2,3,4,5], "bloodInput": "image-derived" }  
}

**End of Chapter 3 (Normative).**

# TIDeS Handbook — Chapter 4

## Time & Dose‑Rate Semantics (Normative Chapter)

**Purpose.** Standardize how time is represented, synchronized, and used to compute pharmacokinetics (PK), **dose‑rate** (D), and **FLASH** flags across heterogeneous systems. This chapter removes ambiguity around clocks, frames, interpolation, numerical integration, and reporting so that independent sites produce **identical** results from the same inputs.

**Audience.** Imaging physicists, dosimetry/PK developers, validator and pipeline engineers, QA leads, and regulators.

**Outcome.** You will: (a) encode injections and acquisitions with unambiguous timestamps, (b) construct analysis frames, (c) fit PK models over explicit windows, (d) compute voxel/ROI **dose‑rate** in Gy/s consistently, (e) determine **FLASH** coverage under versioned policy, and (f) pass all Profile A/B/C timing and dose‑rate rules.

**Normative keywords:** **MUST**, **SHOULD**, **MAY** (RFC 2119/8174).

## 4.0 Time Model: Absolute & Relative (Normative)

1. **Injection epoch.** injectionStart **MUST** be ISO‑8601 with timezone (e.g., 2025-09-25T11:03:27Z). Precision below seconds **MAY** be provided.
2. **Analysis clock.** All analysis times (frames, fit windows) are seconds **relative to** injectionStart (i.e., (t=0) at injectionStart).
3. **Acquisition frames.** frames **MUST** enumerate acquisitions as { tMid\_s, duration\_s }. Frame start = tMid\_s - duration\_s/2; end = tMid\_s + duration\_s/2.
4. **Fitting window.** fitWindow\_s **MAY** specify [t0, t1] (inclusive start, exclusive end) used for PK estimation.
5. **Clock sync.** Known scanner/IS offsets **MUST** be recorded under clockSync and either applied or documented in residualClockSkew\_ms.

**Validator rules:** timing-injection (ERROR), timing-frames (ERROR), timing-fitwindow (WARN if missing where model requires), clock-sync-declared (WARN when offsets known), unit-ucum (ERROR for seconds tokens if mis‑declared).

## 4.1 Canonical Timing Schema (Normative)

{  
 "$id": "https://tides.org/schemas/1.0.0/tides-timings.schema.json",  
 "$schema": "https://json-schema.org/draft/2020-12/schema",  
 "title": "TIDeS Timing",  
 "type": "object",  
 "required": ["injectionStart","frames"],  
 "properties": {  
 "injectionStart": {"type": "string", "format": "date-time"},  
 "frames": {  
 "type": "array",  
 "items": {  
 "type": "object",  
 "required": ["tMid\_s","duration\_s"],  
 "properties": {  
 "tMid\_s": {"type": "number", "minimum": 0},  
 "duration\_s": {"type": "number", "exclusiveMinimum": 0}  
 }  
 },  
 "minItems": 1  
 },  
 "fitWindow\_s": {  
 "type": "array",  
 "items": {"type": "number", "minimum": 0},  
 "minItems": 2,  
 "maxItems": 2  
 },  
 "rawFrames": {"type": "array"},  
 "clockSync": {  
 "type": "object",  
 "properties": {  
 "scannerMinusIS\_ms": {"type": "number"},  
 "applied": {"type": "boolean"},  
 "residualClockSkew\_ms": {"type": "number"}  
 }  
 }  
 }  
}

## 4.2 Sampling Adequacy & Frame Design (Normative)

* **Mono‑exponential PK**: **SHOULD** have **≥3** frames spanning uptake and clearance (e.g., (~10,min), 1 h, 6–24 h).
* **Bi‑exponential PK**: **SHOULD** have **≥4** frames, including at least two late points to constrain the slow component.
* **Dynamic sequences**: If sub‑minute bins are reconstructed, you **MAY** aggregate for PK; preserve original bins in rawFrames.

**Validator rule:** sampling-adequacy (WARN) with model‑specific minima.

## 4.3 From Counts to Activity vs Time (A(t))

**Given:** PET/SPECT dynamic or multi‑timepoint data with decay correction policy and calibration factors (§5.1).

**Activity extraction (normative steps):** 1. Determine ROI/voxel signal per frame. 2. Undo/redo decay correction as needed to a common reference (injectionStart or frame mid) per declared policy. 3. Convert to **activity** in Bq using scanner factor and cross‑cal factors. 4. Associate each frame with (t\_{mid}) and (t).

**Audit fields (MUST):** decayCorrectionPolicy, halfLife\_s, scannerFactor\_Bq\_per\_count, crossCal, acquisitionCorrections.

## 4.4 PK Windows, Interpolation & Integration (Normative + Algorithms)

### 4.4.1 Interpolation Policy

* Within frames, intensity **MAY** be assumed rectangular (uniform over frame) for simple trapezoids.
* Between frames, intensity **SHOULD** be linearly interpolated unless a model fit is performed.

### 4.4.2 TIA Integration

* **TIA** (Bq·s) per voxel/ROI is the trapezoid of measured frames within [t0, t1] plus analytic tail from the fitted model beyond the last frame; see Chapter 5.

**Reference pseudocode (normative):**

function trapezoid\_tia(frames):  
 # frames: list of (t\_mid, dur, A\_mid)  
 sort by t\_mid  
 tia = 0  
 for i in 1..N:  
 t0 = t\_mid[i] - dur[i]/2  
 t1 = t\_mid[i] + dur[i]/2  
 # rectangle over frame  
 tia += A\_mid[i] \* dur[i]  
 # add trapezoid to next frame start (linear interp)  
 if i < N:  
 next\_t0 = t\_mid[i+1] - dur[i+1]/2  
 dt = max(0, next\_t0 - t1)  
 if dt > 0:  
 tia += 0.5 \* (A\_mid[i] + A\_mid[i+1]) \* dt  
 return tia

## 4.5 Dose & Dose‑Rate Definitions (Normative)

* **Absorbed dose** (D()) in Gy is energy per mass delivered to voxel ().
* **Dose‑rate** (D(,t)) in Gy/s is the time derivative of dose at ().

Two equivalent computation pathways are permitted; the chosen pathway **MUST** be declared.

### 4.5.1 Path A — Convolution from Source Kinetics (Preferred when kernels are time‑local)

Given source activity kernel (k(s,)) with units Gy/(Bq·s), and source activity (A\_s(t)): [ D(,t) = \_s k(s,0) \* A\_s(t) ] with appropriate discretization. For voxel self‑dose, S‑values or MC kernels define (k).

### 4.5.2 Path B — Temporal Differentiation of Cumulative Dose

If cumulative dose grids (D(; t\_i)) are available at sub‑second steps (e.g., external beam), (D) **MAY** be approximated by finite differences. For theranostics this is rare; default to **Path A**.

**Validator rule:** dose-rate-method-declared (WARN) and unit-doserate (ERROR for non‑Gy/s).

## 4.6 FLASH Policy (Normative)

**Default policy (version 1.0.0):** A voxel is flash=true when instantaneous dose‑rate ≥ **40 Gy/s** for ≥ **1 ms** (see Chapter 1).

**Policy pack requirements:** - { name: policy-flash, version: 1.0.0, specMajor: 1, thresholds: { doseRate\_Gy\_per\_s: 40, window\_ms: 1 } } in provenance.policyPacks. - Sites **MAY** override thresholds with a **versioned** pack; validator verifies specMajor compatibility.

**Reporting:** flashCoverage\_pc globally and per ROI in TidesDoseMap/TidesDoseReport.

**Validator rules:** flash-policy-version (ERROR), flash-coverage (WARN if missing percentages), unit-doserate (ERROR).

## 4.7 Numerical Reference Implementations (Exhaustive)

These are **reference** snippets to ensure cross‑implementation agreement. Optimize as needed, but preserve semantics.

### 4.7.1 Python — Frame Construction & PK Windowing

from dataclasses import dataclass  
from typing import List, Optional, Tuple  
  
@dataclass  
class Frame:  
 t\_mid\_s: float  
 duration\_s: float  
 A\_mid\_Bq: float # activity at frame midpoint  
  
@dataclass  
class Timing:  
 injectionStart: str # ISO‑8601  
 frames: List[Frame]  
 fitWindow\_s: Optional[Tuple[float, float]] = None  
  
 def sorted\_frames(self) -> List[Frame]:  
 return sorted(self.frames, key=lambda f: f.t\_mid\_s)  
  
 def check\_sampling(self, model: str) -> Tuple[bool, str]:  
 n = len(self.frames)  
 if model == 'monoexp' and n < 3:  
 return False, 'Monoexp SHOULD have ≥3 frames'  
 if model == 'biexp' and n < 4:  
 return False, 'Biexp SHOULD have ≥4 frames'  
 return True, 'OK'

### 4.7.2 Python — TIA (Trapezoid + Analytic Tail)

def tia\_trapezoid(frames: List[Frame]) -> float:  
 fr = sorted(frames, key=lambda x: x.t\_mid\_s)  
 tia = 0.0  
 for i, f in enumerate(fr):  
 t0 = f.t\_mid\_s - f.duration\_s/2  
 t1 = f.t\_mid\_s + f.duration\_s/2  
 tia += f.A\_mid\_Bq \* f.duration\_s # rectangle over frame  
 if i < len(fr) - 1:  
 next\_t0 = fr[i+1].t\_mid\_s - fr[i+1].duration\_s/2  
 dt = max(0.0, next\_t0 - t1)  
 if dt > 0:  
 tia += 0.5 \* (f.A\_mid\_Bq + fr[i+1].A\_mid\_Bq) \* dt  
 return tia  
  
# Analytic tail for monoexp A(t) = A0 \* exp(-lambda t)  
from math import exp  
  
def tia\_tail\_monoexp(A\_last\_Bq: float, t\_last\_s: float, lam: float) -> float:  
 # Integral from t\_last to infinity of A0\*exp(-lambda t) dt  
 # where A0\*exp(-lambda t\_last) = A\_last  
 return A\_last\_Bq / lam

### 4.7.3 Python — Dose‑Rate from S‑values (Path A)

import numpy as np  
  
def dose\_rate\_from\_svalues(A\_ts: np.ndarray, t\_s: np.ndarray, S\_self\_Gy\_per\_Bq\_s: float) -> np.ndarray:  
 """  
 A\_ts: activity time series for a voxel or ROI [Bq] sampled at times t\_s [s]  
 S\_self\_Gy\_per\_Bq\_s: scalar S‑value for self‑dose (or effective kernel collapse)  
 Returns instantaneous dose‑rate series [Gy/s]  
 """  
 # For pure self‑dose with time‑local kernel, dose‑rate = S \* A(t)  
 return S\_self\_Gy\_per\_Bq\_s \* A\_ts

### 4.7.4 Python — FLASH Coverage (Sliding Window)

def flash\_flags(dose\_rate: np.ndarray, t\_s: np.ndarray, threshold\_Gy\_s: float = 40.0, window\_ms: float = 1.0) -> np.ndarray:  
 win\_s = window\_ms / 1000.0  
 flags = np.zeros\_like(dose\_rate, dtype=bool)  
 j0 = 0  
 for i, t in enumerate(t\_s):  
 # advance j0 to maintain window [t, t+win]  
 while j0 < len(t\_s) and t\_s[j0] < t:  
 j0 += 1  
 # find indices within window  
 j = j0  
 ok = False  
 while j < len(t\_s) and t\_s[j] <= t + win\_s:  
 if dose\_rate[j] < threshold\_Gy\_s:  
 ok = False  
 break  
 ok = True  
 j += 1  
 flags[i] = ok  
 return flags  
  
def flash\_coverage(flags: np.ndarray, roi\_mask: np.ndarray) -> float:  
 # roi\_mask boolean array over voxels; flags per‑voxel OR aggregated externally  
 return 100.0 \* flags[roi\_mask].mean() if roi\_mask.any() else 0.0

### 4.7.5 JavaScript (Node) — Validator Checks for Timing Block

function validateTiming(t) {  
 const errs = [];  
 if (!t.injectionStart || !/Z|[+\-]\d\d:??\d\d$/.test(t.injectionStart)) {  
 errs.push({id: 'timing-injection', severity: 'ERROR', msg: 'ISO‑8601 with timezone required'});  
 }  
 if (!Array.isArray(t.frames) || t.frames.length === 0) {  
 errs.push({id: 'timing-frames', severity: 'ERROR', msg: 'At least one frame required'});  
 } else {  
 t.frames.forEach((f, i) => {  
 if (typeof f.tMid\_s !== 'number' || typeof f.duration\_s !== 'number' || f.duration\_s <= 0) {  
 errs.push({id: 'timing-frames', severity: 'ERROR', msg: `Bad frame ${i}`});  
 }  
 });  
 }  
 if (t.fitWindow\_s && (!Array.isArray(t.fitWindow\_s) || t.fitWindow\_s.length !== 2)) {  
 errs.push({id: 'timing-fitwindow', severity: 'WARN', msg: 'fitWindow\_s must be [t0,t1]'});  
 }  
 return errs;  
}

### 4.7.6 SQL — OMOP View for Dose‑Rate Summaries per ROI

CREATE VIEW tides\_roi\_doserate AS  
SELECT r.study\_uid,  
 d.roi\_code,  
 AVG(dr.dose\_rate\_gys) AS mean\_dose\_rate\_gys,  
 MAX(dr.dose\_rate\_gys) AS max\_dose\_rate\_gys  
FROM tides\_roi\_dose d  
JOIN tides\_doserate\_samples dr  
 ON dr.dosemap\_uri = d.dosemap\_uri AND dr.roi\_code = d.roi\_code  
JOIN tides\_study r ON r.study\_uid = d.study\_uid  
GROUP BY 1,2;

## 4.8 Edge Cases & Resolutions (Normative Guidance)

* **Missed early frame:** If first measurement occurs post‑peak, explicitly document extrapolation method (e.g., assumed bolus at t=0) in params; validator emits WARN: early-missing.
* **Negative times:** Frames with tMid\_s < 0 are **invalid**.
* **Clock drift:** If residualClockSkew\_ms > 500, validator issues WARN: clock-skew; recommend correction.
* **Frame overlaps/gaps:** Overlaps **MAY** be allowed (dynamic recon); gaps are handled by trapezoid interpolation; both are recorded in QA logs.

**Validator rules:** frame-overlap (INFO), frame-gap (INFO), clock-skew (WARN).

## 4.9 Reporting: Dose‑Rate & FLASH in Dose Reports

* TidesDoseReport.series[\*].doseRateSummary **SHOULD** include { Ddot\_max\_Gy\_per\_s, Ddot\_p95\_Gy\_per\_s, flashCoverage\_pc } globally and per ROI.
* If dose‑rate not computed, field **MAY** be omitted for Profile C; Profiles A/B **SHOULD** include it when kernels permit.

**YAML excerpt:**

doseRateSummary:  
 global: { Ddot\_max\_Gy\_per\_s: 52.4, Ddot\_p95\_Gy\_per\_s: 8.7, flashCoverage\_pc: 3.1 }  
 byROI:  
 - { roiCode: SNOMED:74281007, Ddot\_max\_Gy\_per\_s: 23.1, flashCoverage\_pc: 1.2 }

## 4.10 Performance & Precision (Implementation Notes)

* **Time grids:** Use monotonic arrays in seconds; avoid accumulating error by repeatedly summing frame durations. Store absolute epoch once; keep floats in seconds thereafter.
* **Precision:** float64 recommended for PK/dose‑rate steps to minimize rounding.
* **Vectorization:** For voxelwise (D) with S‑values, exploit that (D = S A(t)) to avoid per‑voxel convolutions when kernels are diagonal/self‑dominant.

## 4.11 Must‑Support Matrix (Profiles A/B/C)

| Capability | A (Clinical‑Full) | B (Research‑Voxel) | C (Legacy‑Organ) |
| --- | --- | --- | --- |
| injectionStart ISO‑8601 | **MUST** | **MUST** | **MUST** |
| frames[{tMid\_s,duration\_s}] | **MUST** | **MUST** | **SHOULD** |
| fitWindow\_s | **SHOULD** | **SHOULD** | **MAY** |
| Dose‑rate in Gy/s | **SHOULD** | **MUST** (if kernels provided) | **MAY** |
| FLASH policy declared | **MUST** | **MUST** | **MAY** |
| Dose‑rate summary in report | **SHOULD** | **SHOULD** | **MAY** |

## 4.12 Validator Rule Set (Timing & Dose‑Rate Layer)

* timing-injection — injectionStart ISO‑8601 with tz (ERROR)
* timing-frames — valid frames (ERROR)
* sampling-adequacy — model‑aware minima (WARN)
* timing-fitwindow — fit window present where required (WARN)
* clock-sync-declared — known offsets recorded (WARN)
* dose-rate-method-declared — path A/B specified (WARN)
* unit-doserate — Gy/s only (ERROR)
* flash-policy-version — versioned policy or default (ERROR)
* flash-coverage — coverage metrics (WARN)

## 4.13 TIDeS‑CHK‑4 — Time & Dose‑Rate Readiness

* injectionStart ISO‑8601 with timezone.
* Frames list with {tMid\_s,duration\_s}; no negative times.
* Fit window defined and justified (if used).
* Calibration and decay correction policy documented.
* PK windowing and interpolation policy stated.
* Dose‑rate method declared; units in Gy/s.
* FLASH policy packed & versioned; coverage computed.
* Dose‑rate summaries present in TidesDoseReport (A/B).

## 4.14 Traceability (Extract)

4.0,timing-injection,/injectionStart,case\_pass\_minimal  
4.1,timing-frames,/frames,case\_sampling\_inadequate  
4.5,unit-doserate,/doseRateSummary,case\_flash\_flag\_coverage  
4.6,flash-policy-version,/provenance/policyPacks[\*],case\_flash\_flag\_coverage

## 4.15 FAQs

**Q: Our PET console stores local time without timezone.**  
A: Convert at export; record original clock and offset under clockSync. Without timezone, validator blocks (ERROR).

**Q: Can we approximate dose‑rate from two cumulative dose maps?**  
A: Yes for modalities producing time‑resolved dose, but theranostics typically lack such grids; prefer Path A with kernels.

**Q: How do we handle infusion (non‑bolus) administrations?**  
A: Model the input function explicitly (boxcar or measured infusion curve) and ensure frames and PK windows reflect infusion start/stop.

## 4.16 Chapter Summary

* Time is **absolute at injection**, **relative in analysis**. Frames are explicit.
* Dose‑rate is **Gy/s** with a declared computation path.
* FLASH coverage is computed under a **versioned** policy and reported per ROI.
* The validator operationalizes these rules to guarantee cross‑site reproducibility.

**End of Chapter 4 (Normative).**

# TIDeS Handbook — Chapter 5

## PK & Dosimetry (Computable, Normative Chapter)

**Purpose.** Specify the end‑to‑end computational pathway from time‑activity data and calibrations to **time‑integrated activity (TIA)**, **absorbed dose (D)**, and **dose‑rate ((D))**, including models, uncertainty, and reporting—so two independent implementations produce the **same numbers** from the same inputs.

**Audience.** Clinical physicists, imaging/PK & dosimetry developers, QA leads, vendors, and regulators.

**Outcome.** You will (a) calibrate activity correctly, (b) fit PK models with reproducible algorithms and constraints, (c) integrate TIA with analytic tails, (d) compute voxel and ROI dose via S‑values or MC kernels, (e) apply optional corrections (PV, HU→ρ) in a controlled manner, (f) quantify uncertainty, and (g) generate standards‑compliant dose reports that pass Profile A/B/C.

**Normative keywords:** **MUST**, **SHOULD**, **MAY**.

## 5.0 Scope & Invariants (Normative)

1. **Units** **MUST** be UCUM (Chapter 1): Bq, Bq·s, Gy, Gy/s, s.
2. **Inputs** are lossless DICOM (Chapter 3) with declared corrections and **FoR**.
3. **Time model** uses injectionStart and explicit frames (Chapter 4).
4. **Provenance** and **uncertainty** are mandatory for key metrics.

Validator anchors: unit-ucum, timing-injection, sampling-adequacy, kernel-metadata-declared, uncertainty-declared, provenance.

## 5.1 Calibration & Activity Recovery (Normative)

### 5.1.1 Required Inputs

* **Scanner factor** scannerFactor\_Bq\_per\_count (**MUST**)
* **Decay correction policy** (**MUST**) and **half‑life** halfLife\_s (**MUST**)
* **Cross‑calibration** to dose calibrator (factor, date, device) (**SHOULD**)
* **Acquisition corrections** flags for attenuation, scatter, randoms, normalization, dead‑time (**MUST**)

### 5.1.2 Activity Computation

1. Extract ROI/voxel signal per frame.
2. Adjust for decay to chosen reference (frame mid or injection start) per policy.
3. Multiply by scanner and cross‑cal factors → A\_mid\_Bq.
4. Associate (tMid\_s, duration\_s, A\_mid\_Bq) for each frame.

**Validator rules:** dicom-corrections-declared (ERROR), calibration-present (WARN), half-life-mismatch (WARN).

## 5.2 Pharmacokinetic (PK) Models (Normative)

TIDeS supports **parametric** (mono‑exponential, bi‑exponential, two‑compartment) and **non‑parametric** (spline) models. Implementations **MUST** declare model, weights, initial guesses, constraints, and fit method.

### 5.2.1 Mono‑Exponential (monoexp)

* **Model:** ( A(t) = A\_0,e^{-t} )
* **Params:** A0 > 0, λ > 0.
* **Weights:** **SHOULD** be inverse‑variance; WLS recommended.
* **TIA tail:** (\_ = A(t\_L)/) where (t\_L) is last time, (A(t\_L)) fitted.

### 5.2.2 Bi‑Exponential (biexp)

* **Model:** ( A(t) = A\_1 e^{-\_1 t} + A\_2 e^{-\_2 t} )
* **Constraints:** A1 ≥ 0, A2 ≥ 0, λ1 > λ2 > 0 (fast then slow).
* **TIA tail:** ( A\_1/\_1 + A\_2/\_2 - \_0^{t\_L} A(t) , dt ) if integrating from 0→∞; in practice: trapezoid up to (t\_L) + analytical beyond (t\_L) using fitted params.

### 5.2.3 Two‑Compartment (2‑Comp)

* **Model:** Blood‑tissue system with ODEs (e.g., reversible or irreversible uptake). Example irreversible model with plasma input (C\_p(t)): [ = K\_1 C\_p(t) - (k\_2+k\_3) C\_t(t) ] Observed activity (A(t) = V\_t C\_t(t)) scaled by voxel/ROI volume.
* **Params:** K1, k2, k3 (and optional blood volume fraction Vb).
* **Input:** arterial or image‑derived input function (IDIF); smoothing and delay **MUST** be documented.

### 5.2.4 Spline / Non‑Parametric

* **Model:** Constrained cubic splines through measured mid‑points; monotonic non‑negative tail enforced.
* **Tail:** exponential extrapolation from last two knots with slope ≤ 0.

**Validator rules:** model-declared (ERROR), sampling-adequacy (WARN), covariance-present (WARN when delta‑method used), priors-declared (INFO for Bayesian fits).

## 5.3 Fitting Methods & Diagnostics (Normative)

* **Optimization:** Nonlinear least squares with bounds (e.g., Trust Region Reflective) **SHOULD** be used; convergence criteria and tolerances **MUST** be recorded.
* **Weights:** If WLS, show how variances were estimated (e.g., Poisson counts → delta‑method).
* **Initial guesses:** Document deterministic seeding rules; avoid random seeds for clinical reproducibility.
* **Diagnostics:** Report **R²**, **AIC**/**BIC**, residuals (mean, RMS), and parameter covariance matrix.
* **Failure modes:** If fit fails or is non‑identifiable, demote the model (e.g., monoexp fallback) and annotate in provenance.

**Validator rules:** fit-diagnostics-present (WARN), fit-bounds-satisfied (ERROR if constraints violated).

## 5.4 Time‑Integrated Activity (TIA) (Normative)

**Definition:** ( = *0^{} A(t),dt =* {} + \_{})

**Measured part:** trapezoidal integration across frames with rectangular in‑frame assumption (Chapter 4 pseudocode is normative).

**Analytic tails:** - **monoexp:** ( A(t\_L)/) - **biexp:** ( A\_1/\_1 + A\_2/\_2 - \_0^{t\_L} A(t),dt ) - **2‑Comp/Spline:** integrate fitted function beyond (t\_L) with non‑negative decay.

**Validator rules:** tia-method-declared (ERROR), unit-ucum on Bq·s (ERROR).

## 5.5 Voxel Dose — S‑Values & MC Kernels (Normative)

### 5.5.1 S‑Value Formalism

* **Dose at voxel x:** ( D() = \_s S( s), (s) )
* **Kernel metadata (MUST):** { nuclide, medium, grid\_mm, support\_vox, formalism, version, sourceURI }
* **Boundary condition:** Zero‑padding unless periodic convolution explicitly declared (not recommended).

### 5.5.2 Monte Carlo (MC) Kernels

* **Kernel:** 3D energy deposition per source voxel normalized per decay; grid and medium **MUST** be declared.
* **Heterogeneity:** If HU→ρ scaling is used, document mapping and energy dependence.

### 5.5.3 Implementation Choices

* **FFT‑based convolution** for uniform kernels and grids;
* **Direct sparse stencil** for compact support S‑values;
* **Chunked tiling** for large volumes.

**Validator rules:** kernel-metadata-declared (ERROR), kernel-grid-compat (ERROR if mismatch), unit-dose-gy (ERROR).

## 5.6 Dose‑Rate (Link to Chapter 4)

* With time‑local kernels, ( D(,t) = S\_{self},A(,t) ) for self‑dose approximation.
* For cross‑dose, convolve instantaneous activity with kernel at each time step (computationally heavy; **MAY** approximate when cross‑dose fraction is negligible for the nuclide and organ).

**Validator rules:** dose-rate-method-declared (WARN), unit-doserate (ERROR).

## 5.7 Optional Corrections (Controlled)

### 5.7.1 HU→ρ (Density)

* **MAY** be applied for heterogeneity corrections; mapping **MUST** be declared (points or function); if omitted, assume soft tissue default.

### 5.7.2 Partial Volume (PV)

* **MAY** be applied.
  + **RC‑based** (recovery coefficients vs size): document PSF (FWHM), object size assumptions.
  + **PSF deconvolution**: record kernel, iterations, regularization.
* **Order**: Apply PV correction **before** PK fitting when correcting measured activities.

**Validator rules:** pv-method-declared (WARN), hu-density-declared (WARN).

## 5.8 Uncertainty Propagation (Normative)

### 5.8.1 Methods

* **Delta‑method:** propagate parameter covariance through TIA and dose.
* **Bootstrap:** resample frames (with replacement), refit, recompute TIA and dose for ≥1000 iterations (**SHOULD**), report 68% CI half‑width as 1‑σ.
* **Calibration & PV components:** include as Type B; combine in quadrature.

### 5.8.2 Delta‑Method Jacobians (Canonical)

* **monoexp:** ( A(t) = A\_0 e^{-t} ), ( = A dt = A\_0/)
  + ( = 1/), ( = -A\_0/^2 )
* **biexp:** ( = A\_1/\_1 + A\_2/\_2 )

**Validator rules:** uncertainty-declared (ERROR), uncertainty-method-known (WARN), covariance-present (WARN if delta‑method chosen but covariance missing).

## 5.9 Organ Summaries & DVH (Normative)

Per ROI, report **Dmean\_Gy**, **Dmax\_Gy**, optional **Dx\_Gy** (dose to x% volume) and **Vx** (fraction above x Gy). Units **MUST** be UCUM. DVH **SHOULD** be computed on the dose grid linked to the FoR.

**Validator rules:** unit-dose-gy (ERROR), roi-coded (ERROR/WARN per profile).

## 5.10 Reference Implementations (Exhaustive Code)

The following code is reference‑quality and favors clarity and reproducibility over micro‑optimizations. You **MAY** optimize but **MUST** preserve outputs within numerical tolerances.

### 5.10.1 Python — PK Fitting (Monoexp & Biexp, WLS)

from dataclasses import dataclass  
from typing import List, Tuple, Dict  
import numpy as np  
from numpy.linalg import lstsq  
from math import log  
  
@dataclass  
class Frame:  
 t\_mid\_s: float  
 duration\_s: float  
 A\_mid\_Bq: float  
  
@dataclass  
class FitResult:  
 params: Dict[str, float]  
 cov: np.ndarray # parameter covariance  
 r2: float  
 aic: float  
 bic: float  
 residuals: np.ndarray  
  
def wls\_monoexp(frames: List[Frame], weights: np.ndarray = None) -> FitResult:  
 """Weighted log-linear fit with bias correction for finite frame duration."""  
 t = np.array([f.t\_mid\_s for f in frames])  
 y = np.array([max(f.A\_mid\_Bq, 1e-12) for f in frames])  
 # Optional: correct for within-frame averaging if durations are long  
 X = np.vstack([np.ones\_like(t), -t]).T # log A = log A0 - lambda t  
 W = np.eye(len(t)) if weights is None else np.diag(weights)  
 beta, \*\_ = lstsq(W @ X, W @ np.log(y), rcond=None)  
 logA0, lam = beta[0], beta[1]  
 A0 = np.exp(logA0)  
 # Residuals  
 yhat = A0 \* np.exp(-lam \* t)  
 resid = y - yhat  
 # Covariance (OLS/WLS approx)  
 sigma2 = (resid @ (W @ resid)) / max(len(t) - 2, 1)  
 cov = sigma2 \* np.linalg.inv(X.T @ W @ X)  
 # Metrics  
 ss\_res = np.sum(resid\*\*2)  
 ss\_tot = np.sum((y - y.mean())\*\*2)  
 r2 = 1 - ss\_res/ss\_tot if ss\_tot > 0 else 1.0  
 n = len(t)  
 k = 2  
 aic = n \* np.log(ss\_res/n + 1e-30) + 2\*k  
 bic = n \* np.log(ss\_res/n + 1e-30) + k\*np.log(n)  
 return FitResult(params={"A0": A0, "lambda": lam}, cov=cov, r2=r2, aic=aic, bic=bic, residuals=resid)  
  
from scipy.optimize import least\_squares  
  
def wls\_biexp(frames: List[Frame], w: np.ndarray = None) -> FitResult:  
 t = np.array([f.t\_mid\_s for f in frames])  
 y = np.array([max(f.A\_mid\_Bq, 1e-12) for f in frames])  
 if w is None:  
 w = np.ones\_like(t)  
 # Initial guesses: split slope by linear fit, small slow component  
 mono = wls\_monoexp(frames)  
 A1\_0 = 0.7\*mono.params['A0']  
 A2\_0 = 0.3\*mono.params['A0']  
 lam1\_0 = mono.params['lambda'] \* 2.0  
 lam2\_0 = mono.params['lambda'] \* 0.2  
  
 def model(p):  
 A1, A2, lam1, lam2 = p  
 return A1\*np.exp(-lam1\*t) + A2\*np.exp(-lam2\*t)  
  
 def resid(p):  
 A1, A2, lam1, lam2 = p  
 if (A1 < 0) or (A2 < 0) or (lam1 <= lam2) or (lam2 <= 0):  
 return 1e6\*np.ones\_like(t)  
 return (y - model(p)) \* np.sqrt(w)  
  
 p0 = np.array([A1\_0, A2\_0, lam1\_0, lam2\_0])  
 ls = least\_squares(resid, p0, method='trf')  
 A1, A2, lam1, lam2 = ls.x  
 yhat = model(ls.x)  
 resid\_vec = y - yhat  
 # Covariance via Jacobian  
 J = ls.jac  
 dof = max(len(t) - 4, 1)  
 sigma2 = (resid\_vec @ resid\_vec) / dof  
 cov = sigma2 \* np.linalg.inv(J.T @ J)  
 ss\_res = np.sum(resid\_vec\*\*2)  
 ss\_tot = np.sum((y - y.mean())\*\*2)  
 r2 = 1 - ss\_res/ss\_tot if ss\_tot > 0 else 1.0  
 n = len(t); k = 4  
 aic = n\*np.log(ss\_res/n + 1e-30) + 2\*k  
 bic = n\*np.log(ss\_res/n + 1e-30) + k\*np.log(n)  
 return FitResult(params={"A1":A1,"A2":A2,"lambda1":lam1,"lambda2":lam2}, cov=cov, r2=r2, aic=aic, bic=bic, residuals=resid\_vec)

### 5.10.2 Python — TIA (Trapezoid + Analytic Tails)

def tia\_trapezoid(frames: List[Frame]) -> float:  
 fr = sorted(frames, key=lambda x: x.t\_mid\_s)  
 tia = 0.0  
 for i, f in enumerate(fr):  
 t0 = f.t\_mid\_s - f.duration\_s/2  
 t1 = f.t\_mid\_s + f.duration\_s/2  
 tia += f.A\_mid\_Bq \* f.duration\_s  
 if i < len(fr) - 1:  
 next\_t0 = fr[i+1].t\_mid\_s - fr[i+1].duration\_s/2  
 gap = max(0.0, next\_t0 - t1)  
 if gap > 0:  
 tia += 0.5 \* (f.A\_mid\_Bq + fr[i+1].A\_mid\_Bq) \* gap  
 return tia  
  
def tia\_tail\_mono(lam: float, A\_last: float) -> float:  
 return A\_last / lam  
  
def tia\_tail\_bi(A1: float, lam1: float, A2: float, lam2: float, t\_last: float) -> float:  
 # full analytic integral from t\_last to infinity  
 return A1\*np.exp(-lam1\*t\_last)/lam1 + A2\*np.exp(-lam2\*t\_last)/lam2

### 5.10.3 Python — Voxel Dose via FFT Convolution (S‑value Kernel)

import numpy as np  
from numpy.fft import rfftn, irfftn  
  
def pad\_to\_next\_pow2(shape):  
 return tuple(1<< (int(s-1).bit\_length()) for s in shape)  
  
def convolve3d\_fft(tia: np.ndarray, kernel: np.ndarray) -> np.ndarray:  
 """Convolve TIA [Bq·s] with S‑value kernel [Gy/(Bq·s)] → Dose [Gy].  
 tia and kernel on same grid; zero‑pad; periodic wrap avoided by padding.  
 """  
 assert tia.shape == kernel.shape  
 pad\_shape = pad\_to\_next\_pow2(tia.shape)  
 pad = [(0, p - s) for s, p in zip(tia.shape, pad\_shape)]  
 T = np.pad(tia, pad, mode='constant')  
 K = np.pad(kernel, pad, mode='constant')  
 F\_T = rfftn(T)  
 F\_K = rfftn(K)  
 F\_D = F\_T \* F\_K  
 D = irfftn(F\_D, s=pad\_shape)  
 # crop to original  
 return D[tuple(slice(0, s) for s in tia.shape)]

### 5.10.4 Python — DVH & Organ Metrics

def dvh\_metrics(dose\_gy: np.ndarray, roi\_mask: np.ndarray, x\_gy: float) -> dict:  
 vox = dose\_gy[roi\_mask]  
 if vox.size == 0:  
 return {"Dmean\_Gy": 0.0, "Dmax\_Gy": 0.0, "Vx": {f"{x\_gy}Gy": 0.0}}  
 dmean = float(vox.mean())  
 dmax = float(vox.max())  
 vx = float((vox >= x\_gy).mean())  
 return {"Dmean\_Gy": dmean, "Dmax\_Gy": dmax, "Vx": {f"{x\_gy}Gy": vx}}

### 5.10.5 Python — Bootstrap Uncertainty

rng = np.random.default\_rng(42)  
  
def bootstrap\_tia\_dose(frames: List[Frame], kernel\_self: float, n=1000):  
 t = np.array([f.t\_mid\_s for f in frames])  
 A = np.array([f.A\_mid\_Bq for f in frames])  
 dur = np.array([f.duration\_s for f in frames])  
 tls = []  
 for \_ in range(n):  
 idx = rng.integers(0, len(frames), len(frames))  
 fr\_b = [Frame(t[i], dur[i], A[i]) for i in idx]  
 tia = tia\_trapezoid(fr\_b)  
 dose = kernel\_self \* tia # self‑dose scalar example  
 tls.append((tia, dose))  
 arr = np.array(tls)  
 tia\_sd = float(np.std(arr[:,0], ddof=1))  
 dose\_sd = float(np.std(arr[:,1], ddof=1))  
 return {"tia\_sd": tia\_sd, "dose\_sd": dose\_sd}

### 5.10.6 Python — Delta‑Method for Monoexp TIA

def delta\_method\_tia\_mono(cov: np.ndarray, A0: float, lam: float) -> float:  
 # J = [∂TIA/∂A0, ∂TIA/∂λ] = [1/λ, −A0/λ²]  
 J = np.array([1.0/lam, -A0/(lam\*\*2)])  
 var = float(J @ cov @ J.T)  
 return np.sqrt(max(var, 0.0))

### 5.10.7 Two‑Compartment (Irreversible) — Numerical ODE (Python)

from math import isfinite  
  
def integrate\_two\_comp(t\_s: np.ndarray, Cp: np.ndarray, K1: float, k2: float, k3: float, C0: float = 0.0) -> np.ndarray:  
 """Simple forward Euler / RK4 for C\_t'(t) = K1\*Cp - (k2+k3)\*C\_t.  
 t\_s ascending; Cp aligned; returns C\_t at t\_s.  
 """  
 Ct = np.zeros\_like(t\_s)  
 Ct[0] = C0  
 for i in range(1, len(t\_s)):  
 dt = t\_s[i] - t\_s[i-1]  
 # RK4  
 def f(ct, cp):  
 return K1\*cp - (k2+k3)\*ct  
 k1 = f(Ct[i-1], Cp[i-1])  
 k2v = f(Ct[i-1] + 0.5\*dt\*k1, 0.5\*(Cp[i-1]+Cp[i]))  
 k3v = k2v  
 k4 = f(Ct[i-1] + dt\*k3v, Cp[i])  
 Ct[i] = Ct[i-1] + (dt/6.0)\*(k1 + 2\*k2v + 2\*k3v + k4)  
 if not isfinite(Ct[i]):  
 raise ValueError('Integration blew up')  
 return Ct

## 5.11 CLI/API Contracts (Executable)

### 5.11.1 CLI

# PK fit  
$ tides pk-fit --model monoexp --frames frames.json --out pk.json \  
 --weights poisson --seed deterministic  
  
# TIA & dose  
$ tides dose --pk pk.json --kernel svalue\_177Lu\_v1.npz --out dose.nii.gz \  
 --method fft --report report.json  
  
# Bootstrap  
$ tides uncertainty bootstrap --frames frames.json --kernel svalue\_177Lu\_v1.npz \  
 --iters 1000 --out unc.json

### 5.11.2 HTTP API (OpenAPI 3.1 extract)

post: /pk/fit  
requestBody:  
 application/json:  
 schema: { $ref: '#/components/schemas/Frames' }  
responses:  
 '200': { description: FitResult }  
  
post: /dose/compute  
requestBody:  
 application/json:  
 schema: { type: object, properties: { tiaUri: {type: string}, kernel: {type: object} } }  
responses:  
 '200': { description: Dose map URI }

## 5.12 Profiles Must‑Support (A/B/C)

| Capability | A (Clinical‑Full) | B (Research‑Voxel) | C (Legacy‑Organ) |
| --- | --- | --- | --- |
| Declared model & weights | **MUST** | **MUST** | **SHOULD** |
| TIA trapezoid + tail | **MUST** | **MUST** | **MUST** |
| Kernel metadata | **MUST** | **MUST** | **N/A** |
| Voxel dose (RTDOSE) | **MUST** | **MUST** | **MAY** |
| Uncertainty (method + %) | **MUST** | **MUST** | **MUST** |
| Fit diagnostics (R²/AIC) | **SHOULD** | **SHOULD** | **MAY** |
| PV/HU→ρ declaration | **SHOULD** | **SHOULD** | **MAY** |

## 5.13 Validator Rule Set (PK & Dosimetry Layer)

* model-declared — PK model and parameters present (ERROR)
* sampling-adequacy — frames meet minima (WARN)
* tia-method-declared — trapezoid + tail specified (ERROR)
* kernel-metadata-declared — nuclide/medium/grid/support/version (ERROR)
* kernel-grid-compat — kernel and dose grid compat (ERROR)
* pv-method-declared — PV correction documented (WARN)
* hu-density-declared — HU→ρ mapping present if used (WARN)
* fit-diagnostics-present — R²/AIC/BIC/residuals/covariance (WARN)
* uncertainty-declared — % and method present (ERROR)

## 5.14 Worked Example (End‑to‑End)

**Narrative:** 177Lu therapy; monoexp ROI PK; TIA via trapezoid + tail; dose via S‑values; bootstrap uncertainty.

**YAML summary:**

pkModel:  
 id: urn:tides:pkmodel:monoexp:cc33...  
 scope: roi  
 fit: { method: WLS, params: { A0: 1.4e7, lambda: 2.9e-5 }, r2: 0.98, aic: 34.1 }  
 covariance: [[1.2e4, -3.1e-2], [-3.1e-2, 5.4e-8]]  
 tia\_Bq\_s: 4.83e10  
 tail\_method: analytic-monoexp  
  
doseMap:  
 id: urn:tides:dosemap:177Lu:dd44...  
 kernel: { formalism: S-value, nuclide: 177Lu, medium: ICRP-soft, grid: 2mm, support\_vox: 63, version: v1, sourceURI: s3://... }  
 frameOfReferenceUID: 1.2.826...  
  
organSummaries:  
 - roiCode: SNOMED:74281007  
 roiCodeText: liver  
 Dmean\_Gy: 18.2  
 Dmax\_Gy: 21.5  
 Vx: { "10Gy": 0.21 }  
 uncertainty\_pc: 9.1  
 uncertaintyMethod: delta-method  
 qc: { R2: 0.98, AIC: 34.1, residualRMS: 1.2e6 }

## 5.15 QA, Reproducibility & Performance

* **Determinism:** Fix random seeds; log library versions; hash inputs.
* **Tolerances:** Unit tests require relative differences < 1e‑6 for PK and < 1e‑4 for dose maps (FFT padding effects considered).
* **Performance:** Prefer FFT with padding; tile for large volumes; exploit separability if kernel supports it.

## 5.16 TIDeS‑CHK‑5 — PK & Dosimetry Readiness

* Calibration factors and decay policy declared.
* Model, weights, bounds, and initial guesses recorded.
* Frames meet sampling guidance or WARN explains deviation.
* TIA = trapezoid + analytic tail (declared).
* Kernel metadata (nuclide/medium/grid/support/version) present; grid compat verified.
* Optional PV/HU→ρ methods documented.
* Uncertainty (% and method) present; covariance or bootstrap iterations recorded.
* Organ summaries computed with UCUM units; DVH optional.

## 5.17 Traceability (Extract)

5.2,model-declared,/pkModel/fit/method,case\_sampling\_inadequate  
5.4,tia-method-declared,/pkModel/tail\_method,case\_pass\_minimal  
5.5,kernel-metadata-declared,/doseMap/kernel,case\_registration\_ok  
5.8,uncertainty-declared,/organSummaries[\*]/uncertainty\_pc,case\_pass\_minimal

## 5.18 FAQs

**Q: Is biexponential always better than monoexp?**  
A: Not necessarily. Use AIC/BIC and residuals. If data are sparse, monoexp may be more stable; report rationale.

**Q: How do we handle infusion instead of bolus?**  
A: Convolve input with infusion boxcar; fit models that accept non‑bolus input; document infusion start/stop in timings.

**Q: Can we mix kernels from different vendors?**  
A: Only with explicit versioning and validation. Kernels **MUST** match nuclide, grid, and medium; declare sources.

## 5.19 Chapter Summary

* PK models transform counts into **A(t)**; **TIA** integrates A(t) to infinity with an analytic tail; kernels map TIA to dose; uncertainty quantifies trust.
* With declared models, kernels, and methods, TIDeS ensures **reproducible dosimetry** across centers.

### Appendix 5‑A: Closed‑Form Integrals

* **monoexp:** (\_0^{} A\_0 e^{-t} dt = A\_0/)
* **biexp:** (A\_1/\_1 + A\_2/\_2)

### Appendix 5‑B: Parameter Bounds & Guesses (Recommended)

* monoexp: A0 ∈ (0, +∞), λ ∈ (1e-7, 1e-2) s⁻¹
* biexp: λ1 ∈ (1e-5, 1e-2), λ2 ∈ (1e-7, λ1); A1,A2 ≥ 0

### Appendix 5‑C: Kernel Metadata Schema (JSON)

{  
 "formalism": "S-value",  
 "nuclide": "177Lu",  
 "medium": "ICRP-soft",  
 "grid\_mm": [2.0,2.0,2.0],  
 "support\_vox": 63,  
 "version": "v1",  
 "sourceURI": "s3://.../177Lu\_S\_ICRPsoft\_v1.npz"  
}

### Appendix 5‑D: OMOP Tables (Dose Extensions)

CREATE TABLE tides\_doserate\_samples (  
 id BIGSERIAL PRIMARY KEY,  
 dosemap\_uri TEXT NOT NULL,  
 roi\_code TEXT NOT NULL,  
 t\_s DOUBLE PRECISION NOT NULL,  
 dose\_rate\_gys DOUBLE PRECISION NOT NULL  
);

**End of Chapter 5 (Normative).**

# TIDeS Handbook — Chapter 6

## Safety & Constraints (Normative Chapter)

**Purpose.** Define how organ-at-risk (OAR) and protocol safety constraints are represented, versioned, evaluated, and reported for theranostics—consistently across vendors and sites. This chapter specifies the **policy pack format**, **default adult limits**, **pediatric modifiers**, **cumulative dose accounting**, **validator behavior**, and the **reference algorithms and code** that make the rules executable.

**Audience.** Clinical physicists, nuclear medicine physicians, safety officers, software vendors, QA leads, regulators.

**Outcome.** You will (a) install and version a site policy, (b) evaluate per‑series and cumulative OAR limits deterministically, (c) produce machine‑readable safety findings for the validator and the clinical report, and (d) document exceptions with provenance.

**Normative keywords:** **MUST**, **SHOULD**, **MAY** (RFC 2119/8174).

## 6.0 Scope & First Principles (Normative)

1. **Policy‑driven.** Safety checks are driven by **versioned policy packs**. TIDeS ships a default pack (Adult 1.0.0). Sites **MAY** install overrides but **MUST** declare their pack in provenance (provenance.policyPacks).
2. **Deterministic.** Given a policy and a dose report, the evaluation result is fully deterministic and reproducible. No interactive adjustments at evaluation time.
3. **Hierarchical.** Limits may exist at **per‑series** (current treatment), **per‑course** (current admission), and **lifetime/cumulative** levels; precedence rules are explicit.
4. **Explainable.** Every safety decision includes **which rule**, **which value**, **what limit**, **what profile**, and **the evidence** (ROI stats, dose grid reference, dates).
5. **Privacy‑respecting.** Safety logic **MUST NOT** require PHI beyond minimal demographics for pediatric modifiers.

Validator anchors: constraint-fail (ERROR), constraint-warn (WARN), policy-compat (ERROR), policy-declared (ERROR), cumulative-logic (WARN/ERROR), roi-missing (WARN), units-ucum (ERROR).

## 6.1 Default Adult OAR Policy (Normative, v1.0.0)

These limits are the TIDeS **default**; sites **MAY** override by installing their own **versioned** policy pack. All units are UCUM.

| OAR (SNOMED/RadLex exemplar) | Metric | Limit (ERROR) | Pre‑Warn (WARN) | Rationale/Notes |
| --- | --- | --- | --- | --- |
| Kidney (SNOMED:64033007) | Dmean\_Gy | ≤ **23 Gy** | ≥ **20 Gy** | Adult default; per §6.5 cumulative accounted across cycles |
| Liver (SNOMED:74281007) | Dmean\_Gy | ≤ **30 Gy** | ≥ **27 Gy** | Adult default; whole‑liver mean |
| Red marrow (RadLex or SNOMED) | Dmean\_Gy | ≤ **2 Gy** | ≥ **1.8 Gy** | Reference adult limit; marrow mask per site protocol |

**Flash policy:** Not a constraint per se; its computation and reporting are mandatory (Chapter 4). Sites **MAY** add explicit FLASH‑sensitive constraints via policy extensions (see §6.10).

## 6.2 Policy Pack Format (Normative Schema)

### 6.2.1 JSON Schema (authoritative)

{  
 "$id": "https://tides.org/schemas/1.0.0/tides-policy.schema.json",  
 "$schema": "https://json-schema.org/draft/2020-12/schema",  
 "title": "TIDeS Safety Policy Pack",  
 "type": "object",  
 "required": ["name","version","specMajor","oars"],  
 "properties": {  
 "name": {"type": "string"},  
 "version": {"type": "string"},  
 "specMajor": {"type": "integer", "const": 1},  
 "effectiveDate": {"type": "string", "format": "date"},  
 "notes": {"type": "string"},  
 "oars": {  
 "type": "array",  
 "items": {  
 "type": "object",  
 "required": ["roiCode","metric","limit","warnThreshold"],  
 "properties": {  
 "roiCode": {"type": "string"},  
 "roiCodeSystem": {"type": "string", "enum": ["SNOMED","RadLex","LOCAL"]},  
 "roiCodeText": {"type": "string"},  
 "metric": {"type": "string", "enum": ["Dmean\_Gy","Dmax\_Gy","Dx\_Gy","Vx","custom"]},  
 "x": {"type": "number"},  
 "limit": {"type": "number"},  
 "warnThreshold": {"type": "number"},  
 "unit": {"type": "string", "enum": ["Gy","Gy/s","1"]},  
 "aggregation": {"type": "string", "enum": ["per-series","per-course","cumulative"]},  
 "pediatric": {  
 "type": "object",  
 "properties": {  
 "ageBands": {  
 "type": "array",  
 "items": {"type": "object", "properties": {"minY": {"type":"number"}, "maxY": {"type":"number"}, "scale": {"type":"number"}}}  
 },  
 "weightScaling": {"type": "object", "properties": {"refKg": {"type":"number"}, "exponent": {"type":"number"}}}  
 }  
 }  
 }  
 }  
 },  
 "flash": {  
 "type": "object",  
 "properties": {"threshold\_Gy\_per\_s": {"type":"number"}, "window\_ms": {"type":"number"}}  
 }  
 }  
}

### 6.2.2 Example Policy Pack (Default Adult v1.0.0)

name: policy-oar  
version: 1.0.0  
specMajor: 1  
effectiveDate: 2025-01-01  
notes: Default adult limits for TIDeS 1.0  
oars:  
 - { roiCode: "64033007", roiCodeSystem: SNOMED, roiCodeText: kidney, metric: Dmean\_Gy, limit: 23, warnThreshold: 20, unit: Gy, aggregation: cumulative }  
 - { roiCode: "74281007", roiCodeSystem: SNOMED, roiCodeText: liver, metric: Dmean\_Gy, limit: 30, warnThreshold: 27, unit: Gy, aggregation: cumulative }  
 - { roiCode: "marrow", roiCodeSystem: LOCAL, roiCodeText: red marrow, metric: Dmean\_Gy, limit: 2, warnThreshold: 1.8, unit: Gy, aggregation: cumulative }  
flash: { threshold\_Gy\_per\_s: 40, window\_ms: 1 }

**Normative:** specMajor **MUST** match the TIDeS spec’s major version. The validator **MUST** reject policy packs with incompatible specMajor (policy-compat).

## 6.3 Evaluation Semantics (Normative)

1. **Metrics from dose grid.** For each ROI, compute required statistics on the aligned RTDOSE grid (Chapter 3): Dmean\_Gy, Dmax\_Gy, Dx\_Gy, Vx.
2. **Aggregation scope.** Evaluate per‑series **and**, if policy requests, cumulative across TidesStudy lineage (same subject, compatible FoR via stored transforms).
3. **Decision thresholds.**
   * **ERROR/BLOCK** when measured value **exceeds** limit (or fails metric‑specific predicate).
   * **WARN** when measured value **≥ warnThreshold** (pre‑warning band).
   * **INFO** when evaluation deferred due to missing ROI (report roi-missing).
4. **Pediatrics.** When pediatric modifiers are present, transform limits by age/weight scaling before comparison (see §6.4).
5. **Traceability.** Persist a finding block per rule with { ruleId, roi, metric, value, limit, decision, evidence, policy } (§6.8).

## 6.4 Pediatric Modifiers (Normative)

**Age bands:** Each OAR limit may define age‑band scaling scale applied multiplicatively to limit and warnThreshold for subjects whose age (years) falls within [minY, maxY].

**Weight scaling:** Optionally apply allometric scaling scaledLimit = limit \* (weightKg/refKg)^exponent after age‑band scaling. Recommended refKg = 70, exponent ∈ [0.67, 1.0] per site policy.

**Demographics input:** Only { ageYears, weightKg } are required; **MUST NOT** include PHI.

**Example (YAML):**

pediatric:  
 ageBands:  
 - { minY: 0, maxY: 5, scale: 0.5 }  
 - { minY: 6, maxY: 12, scale: 0.7 }  
 - { minY: 13, maxY: 17, scale: 0.85 }  
 weightScaling: { refKg: 70, exponent: 0.75 }

**Validator:** peds-params-valid (ERROR if malformed), peds-applied (INFO: record applied scale).

## 6.5 Cumulative Dose Accounting (Normative)

**Scope:** Cumulative evaluation **SHOULD** sum compatible dose contributions across **TidesStudy** instances for the same subject, respecting registration transforms, FoR, and time windows.

**Methods:** 1. **Voxel‑space accumulation** (preferred for Profiles A/B): deformably map historical dose grids to the current dose FoR and sum voxelwise, then re‑compute ROI statistics. 2. **Organ‑summary accumulation** (Profile C or when voxel dose not available): sum past Dmean\_Gy for the same ROI if segmentation policy is comparable; document method.

**Temporal windows:** Sites **MAY** define rolling windows (e.g., last 12 months) in policy; otherwise cumulative is lifetime‑to‑date.

**Validator:** cumulative-logic (WARN if Profile A/B lacks voxel accumulation while historical grids exist), registration-missing (ERROR if required transform absent).

## 6.6 Reference Algorithms & Code (Executable)

The following reference implementations are deterministic and unit‑testable. Optimize as needed without changing results.

### 6.6.1 Python — Policy Engine & Evaluator

from dataclasses import dataclass  
from typing import List, Optional, Dict, Any  
import numpy as np  
  
@dataclass  
class OARRule:  
 roiCode: str  
 metric: str # 'Dmean\_Gy' | 'Dmax\_Gy' | 'Dx\_Gy' | 'Vx' | 'custom'  
 x: Optional[float]  
 limit: float  
 warn: float  
 unit: str # 'Gy'|'Gy/s'|'1'  
 aggregation: str # 'per-series'|'per-course'|'cumulative'  
 pediatric: Optional[dict] = None  
  
@dataclass  
class Policy:  
 name: str  
 version: str  
 specMajor: int  
 oars: List[OARRule]  
 flash: Optional[dict] = None  
  
@dataclass  
class Finding:  
 ruleId: str  
 roiCode: str  
 metric: str  
 value: float  
 unit: str  
 limit: float  
 warn: float  
 decision: str # 'PASS'|'WARN'|'ERROR'  
 evidence: Dict[str, Any]  
 policy: Dict[str, Any]  
  
  
def scale\_limits(rule: OARRule, ageYears: Optional[float], weightKg: Optional[float]) -> (float, float, Dict[str, float]):  
 L, W = rule.limit, rule.warn  
 applied = {"ageScale": 1.0, "weightScale": 1.0}  
 if rule.pediatric:  
 ped = rule.pediatric  
 if ageYears is not None and 'ageBands' in ped:  
 for band in ped['ageBands']:  
 if band['minY'] <= ageYears <= band['maxY']:  
 s = float(band['scale'])  
 L \*= s; W \*= s; applied['ageScale'] = s  
 break  
 if weightKg is not None and 'weightScaling' in ped:  
 refKg = ped['weightScaling'].get('refKg', 70.0)  
 exp = ped['weightScaling'].get('exponent', 0.75)  
 ws = (weightKg / refKg) \*\* exp  
 L \*= ws; W \*= ws; applied['weightScale'] = ws  
 return L, W, applied  
  
  
def eval\_metric(metric: str, roi\_vals: np.ndarray, x: Optional[float]) -> float:  
 if roi\_vals.size == 0:  
 return float('nan')  
 if metric == 'Dmean\_Gy':  
 return float(roi\_vals.mean())  
 if metric == 'Dmax\_Gy':  
 return float(roi\_vals.max())  
 if metric == 'Dx\_Gy': # dose received by x% volume  
 assert x is not None and 0 < x <= 100  
 k = int(np.ceil((100 - x)/100.0 \* roi\_vals.size))  
 return float(np.partition(roi\_vals, k)[k])  
 if metric == 'Vx': # fraction of volume with dose >= x Gy  
 assert x is not None  
 return float((roi\_vals >= x).mean())  
 raise ValueError('Unknown metric')  
  
  
def evaluate\_oar(rule: OARRule, roi\_vals: np.ndarray, ageYears=None, weightKg=None, evidence\_extra=None) -> Finding:  
 value = eval\_metric(rule.metric, roi\_vals, rule.x)  
 L, W, applied = scale\_limits(rule, ageYears, weightKg)  
 decision = 'PASS'  
 if np.isnan(value):  
 decision = 'INFO'  
 else:  
 if (rule.metric == 'Vx'):  
 # For Vx the limit is a maximum fraction (0..1)  
 if value > L: decision = 'ERROR'  
 elif value >= W: decision = 'WARN'  
 else:  
 if value > L: decision = 'ERROR'  
 elif value >= W: decision = 'WARN'  
 ev = {"n": int(roi\_vals.size), "appliedScales": applied}  
 if evidence\_extra: ev.update(evidence\_extra)  
 return Finding(  
 ruleId=f"oar:{rule.roiCode}:{rule.metric}", roiCode=rule.roiCode, metric=rule.metric,  
 value=float(value) if not np.isnan(value) else value, unit=rule.unit,  
 limit=float(L), warn=float(W), decision=decision,  
 evidence=ev, policy={"name": "policy-oar", "version": "1.0.0"}  
 )

### 6.6.2 Python — Cumulative Accumulation (Voxel‑Space)

import numpy as np  
  
def accumulate\_voxel\_dose(dose\_list: List[np.ndarray], masks: List[np.ndarray], transforms: List[callable]) -> np.ndarray:  
 """Map each historical dose grid into the target FoR using provided transforms,  
 apply ROI mask compatibility if necessary, and sum voxelwise.  
 transforms[i]: function that resamples array i into target grid.  
 """  
 assert len(dose\_list) == len(transforms)  
 acc = np.zeros\_like(dose\_list[-1], dtype=np.float32)  
 for i, D in enumerate(dose\_list):  
 Dt = transforms[i](D)  
 acc += Dt.astype(acc.dtype)  
 return acc

### 6.6.3 JavaScript — Validator Rule Emission

function emitConstraint(findings, profile) {  
 const out = [];  
 findings.forEach(f => {  
 if (f.decision === 'ERROR') {  
 out.push({ id: 'constraint-fail', severity: 'ERROR', desc: `${f.metric} ${f.value} ${f.unit} exceeds limit ${f.limit} for ROI ${f.roiCode}`, profile });  
 } else if (f.decision === 'WARN') {  
 out.push({ id: 'constraint-warn', severity: 'WARN', desc: `${f.metric} ${f.value} near limit ${f.limit} for ROI ${f.roiCode}`, profile });  
 }  
 });  
 return out;  
}

### 6.6.4 SQL — OMOP Views for Safety Monitoring

CREATE VIEW tides\_oar\_flags AS  
SELECT r.study\_uid,  
 d.roi\_code,  
 CASE WHEN d.dmean\_gy > p.limit THEN 'ERROR'  
 WHEN d.dmean\_gy >= p.warn THEN 'WARN'  
 ELSE 'PASS' END AS decision,  
 d.dmean\_gy, p.limit, p.warn  
FROM tides\_roi\_dose d  
JOIN tides\_policy p ON p.roi\_code = d.roi\_code;

## 6.7 Operational Workflow (SOP)

1. **Install policy.** Place the JSON/YAML pack under validator/policy/ and reference in provenance.policyPacks.
2. **Generate dose.** Compute voxel dose and ROI summaries per Chapter 5.
3. **Accumulate.** If cumulative evaluation is configured, fetch prior TidesStudy artifacts for the subject and sum voxelwise (preferred) or organ‑level per policy.
4. **Evaluate.** Run the policy engine across ROIs; collect findings with evidence.
5. **Report.** Embed findings in TidesDoseReport.policyResults and surface human‑readable text in the clinical PDF/HTML (Chapter 12).
6. **Validate.** The reference validator enforces ERROR/WARN outcomes and badge issuance.

## 6.8 Findings Structure (Normative)

**JSON block embedded in TidesDoseReport:**

{  
 "policyResults": {  
 "policy": {"name": "policy-oar", "version": "1.0.0", "specMajor": 1},  
 "findings": [  
 {  
 "ruleId": "oar:64033007:Dmean\_Gy",  
 "roiCode": "64033007",  
 "metric": "Dmean\_Gy",  
 "value": 21.5,  
 "unit": "Gy",  
 "limit": 23.0,  
 "warn": 20.0,  
 "decision": "WARN",  
 "evidence": {"n": 45231, "appliedScales": {"ageScale":1,"weightScale":1}, "doseMap": "urn:tides:dosemap:..."}  
 }  
 ],  
 "summary": {"status": "WARN", "flashCoverage\_pc": 3.1}  
 }  
}

**Normative:** summary.status equals the most severe decision among findings.

## 6.9 FHIR & Reporting Mappings (Normative)

* **FHIR DiagnosticReport**: carry a safety conclusion code (e.g., pass, warn, fail) in conclusionCode with system http://tides.org/codes/safety.
* **FHIR Observation (per ROI)**: represent each OAR metric as an Observation with interpretation (H for high) and references to the DiagnosticReport.
* **FHIR DetectedIssue (optional)**: encode each ERROR as a DetectedIssue with severity=high, evidence pointing to the Observation and policy citation.

## 6.10 Extensibility — Custom & FLASH‑Aware Rules

Sites **MAY** extend policy packs with advanced metrics, e.g.: - **Ddot\_max\_Gy\_per\_s** (max dose‑rate) with limit. - **flashCoverage\_pc** minimum/maximum thresholds. - **Regional constraints** (e.g., sub‑organ zones) by referencing sub‑ROI codes.

**Schema extension pattern:** add metric: "custom" and a customExpr field (e.g., CEL or JSONLogic), and register an interpreter in the validator. The **reference validator** supports customExpr with a sandboxed JSONLogic evaluator (policy-custom-expr).

## 6.11 Profiles Must‑Support (A/B/C)

| Capability | A (Clinical‑Full) | B (Research‑Voxel) | C (Legacy‑Organ) |
| --- | --- | --- | --- |
| Policy pack declared | **MUST** | **SHOULD** | **SHOULD** |
| Default OAR set enforced | **MUST** | **SHOULD** | **SHOULD** |
| Cumulative accounting | **SHOULD** | **SHOULD** | **MAY** |
| Pediatric modifiers | **SHOULD** | **SHOULD** | **MAY** |
| DetectedIssue export (FHIR) | **SHOULD** | **MAY** | **MAY** |

## 6.12 Validator Rule Pack (Safety Layer)

* policy-declared — bundle references a safety policy (ERROR)
* policy-compat — policy specMajor matches spec (ERROR)
* constraint-fail — any OAR limit exceeded (ERROR)
* constraint-warn — within warning band (WARN)
* cumulative-logic — cumulative configured & performed when required (WARN/ERROR)
* roi-missing — ROI absent for a required OAR (WARN)
* units-ucum — UCUM units enforced (ERROR)
* policy-custom-expr — custom metric evaluated or rejected safely (WARN/ERROR on failure)

## 6.13 CI & Fixtures (Executable Evidence)

Provide canonical fixtures (§11): 1. **case\_safety\_violation\_kidney** — Dmean\_Gy=24.1 Gy → constraint-fail (ERROR) 2. **case\_constraint\_warn\_liver** — Dmean\_Gy=28.3 Gy → constraint-warn (WARN) 3. **case\_pediatric\_scaled\_pass** — scaled limit raises headroom → PASS 4. **case\_cumulative\_exceed** — two prior studies push over limit → ERROR 5. **case\_roi\_missing** — missing marrow ROI → roi-missing (WARN)

Each fixture directory **MUST** include: input bundle, policy pack, JSON/text/HTML validator reports, and README.

## 6.14 CLI & API Contracts

### 6.14.1 CLI

$ tides policy show --pack validator/policy/policy-oar-1.0.0.yaml  
$ tides safety eval --bundle examples/case\_\*.json --policy policy-oar-1.0.0.yaml --out report.json  
$ tides safety cumulative --subject urn:tides:subject:... --since 2019-01-01 --out cum.json

### 6.14.2 HTTP API (OpenAPI 3.1 extract)

post: /safety/evaluate  
requestBody:  
 content:  
 application/json:  
 schema:  
 type: object  
 properties:  
 doseReport: { type: object }  
 policy: { type: object }  
responses:  
 '200': { description: Policy findings }

## 6.15 TIDeS‑CHK‑6 — Safety Readiness Checklist

* Policy pack present, versioned, specMajor compatible.
* Required OAR ROIs available or documented as missing.
* ROI metrics computed on aligned RTDOSE grid.
* Pediatric modifiers configured when applicable.
* Cumulative accounting performed per policy.
* Findings embedded in TidesDoseReport and surfaced in clinical report.
* Validator PASS/WARN/ERROR status captured and archived.

## 6.16 Traceability (Extract)

6.1,constraint-fail,/policyResults/findings[\*],case\_safety\_violation\_kidney  
6.2,policy-compat,/provenance/policyPacks[\*]/specMajor,case\_pass\_minimal  
6.5,cumulative-logic,/resources[type=TidesDoseMap],case\_cumulative\_exceed  
6.8,units-ucum,/policyResults/findings[\*]/unit,case\_constraint\_warn\_liver

## 6.17 FAQs

**Q: Our site wants slightly different kidney/liver limits.**  
A: Author a site pack (e.g., policy-oar-1.1.0.yaml) with new numbers, declare it in provenance, and ensure specMajor matches. The validator will enforce your pack.

**Q: We can’t deformably register old dose maps.**  
A: Use organ‑summary accumulation (Profile C method) and document in policy.notes; validator will WARN instead of ERROR for cumulative logic.

**Q: How are exceptions recorded?**  
A: Add a decisionOverride field per finding with { reason, approver, timestamp }. Overrides are **visible** but do not change the validator outcome.

## 6.18 Chapter Summary

* Safety is **policy‑driven, versioned, deterministic, and explainable**.
* Default adult OAR limits are defined, pediatric modifiers supported, cumulative logic standardized.
* Findings are embedded in the dose report and exported to FHIR/OMOP.
* The validator operationalizes all MUSTs via constraint-fail/constraint-warn and policy compatibility checks.

**End of Chapter 6 (Normative).**

# TIDeS Handbook — Chapter 7

## Interoperability Mappings (DICOM ↔ FHIR ↔ OMOP)

**Purpose.** Define **precise, executable** mappings between TIDeS resources and external standards: **DICOM** (imaging & RT objects), **FHIR** (clinical exchange), and **OMOP** (analytics). Provide normative tag lists, profile constraints, transform logic, schemas, and reference code so that exports and imports are **lossless where required**, **traceable**, and **validator‑testable**.

**Audience.** PACS/VNA and RT‑DICOM engineers, FHIR implementers, data platform architects, registry builders, vendors, regulators.

**Outcome.** You will (a) export/import DICOM with correct tags for theranostics, (b) serialize computable results to constrained **FHIR** profiles, (c) store analysis‑ready facts into **OMOP** tables, and (d) pass the TIDeS validator’s interoperability checks for Profiles A/B/C.

**Normative keywords:** **MUST**, **SHOULD**, **MAY** (RFC 2119/8174).

## 7.0 Design Tenets (Normative)

1. **TIDeS is the source of truth.** External representations mirror TIDeS resources without inventing semantics.
2. **Round‑trip fidelity.** DICOM ↔ TIDeS ↔ FHIR/OMOP round‑trips **MUST NOT** lose any **normative** element (units, FoR, timings, kernel metadata, policy decisions, provenance).
3. **Separation of identity vs location.** TIDeS uses URNs for identity and includes dereferenceable URIs only as **attachments**.
4. **Profiles & conformance.** Exported FHIR resources **MUST** declare TIDeS profiles; DICOM objects **MUST** carry required tags; OMOP extensions **MUST** follow the DDL herein.

Validator hooks: dicom-required-tags, rtdose-for-match, seg-coded-meanings, parametric-rwvm, fhir-profile-declared, fhir-unit-ucum, omop-ddl-version, link-referential.

## 7.1 DICOM Mapping (Authoritative)

### 7.1.1 Required IODs & Transfer Syntax

* **IODs:** Enhanced PET, NM, CT, MR; **Segmentation (SEG)**; **Spatial Registration** (rigid/affine/deformable); **RTDOSE**; optional **Parametric Map**.
* **Transfer Syntax:** Inputs for quantitation **MUST** be lossless (see Chapter 3). RTDOSE and SEG **MUST** be lossless.

### 7.1.2 Normative Tag Lists

**Patient/Study/Series (selected):** - (0010,0020) PatientID (de‑identified or coded) - (0020,000D) StudyInstanceUID, (0020,000E) SeriesInstanceUID - (0008,0070) Manufacturer, (0008,1090) ManufacturerModelName

**Geometry:** - (0020,0052) FrameOfReferenceUID **MUST** - (0020,0032) Image Position (Patient), (0020,0037) Image Orientation (Patient)

**PET/NM dose & timing:** - (0018,1072) Radiopharmaceutical Start Time (may be local clock) - (0018,1074) Radionuclide Total Dose (administered) - (0018,1075) Radionuclide Half Life **MUST** - (0054,1000) Series Type, (0054,1201) Number of Time Slices - (0028,0009) Frame Increment Pointer / timing macro as applicable - **Corrections:** (0028,0051) Corrected Image (ATN, SCAT, DEC, NORM, DTIM)

**SEG:** - (0062,0003) Segmented Property Category Code Sequence - (0062,000F) Segmented Property Type Code Sequence - (0062,0001) Segment Number, (0062,0005) Segment Algorithm Name, (0062,0008) Segment Algorithm Type - (5200,9229) Derivation Image Sequence (reference images)

**RTDOSE:** - (3004,000E) Dose Grid Scaling (float scale) - (3004,000C) Grid Frame Offset Vector (Z offsets) - (3004,0001) Dose Units = GY **MUST** - (3006,0010) Referenced Frame of Reference Sequence - (300C,0060) Referenced Structure Set Sequence (optional with RTSTRUCT legacy)

**Parametric Map (optional):** - (0040,9096) Real World Value Mapping Sequence with UCUM codes

**Validator rules:** dicom-required-tags (ERROR), unit-dose-gy (ERROR), parametric-rwvm (ERROR), registration-fuid (ERROR).

### 7.1.3 DICOM → TIDeS Field Map (Extract)

| DICOM Tag | TIDeS Field | Notes |
| --- | --- | --- |
| (0020,0052) FoR UID | frameOfReferenceUID | All spatial objects |
| (0018,1075) Half Life | halfLife\_s | Seconds |
| (0018,1072) Start Time + Study Date/Time | injectionStart | Normalize to ISO‑8601 with tz |
| SEG Type/Category Codes | TidesSeg.roi[ ].code[System] | SNOMED/RadLex |
| RTDOSE Units & Grid | TidesDoseMap.kernel, grid | Units must be Gy |
| RWVM | Parametric map units | UCUM required |

### 7.1.4 TIDeS → DICOM (RTDOSE/SEG) Serialization

* **RTDOSE:** Create dose grid with units GY, populate GridFrameOffsetVector, scaling, FoR reference, and Referenced Image Sequence to parent anatomy.
* **SEG:** For each ROI, encode coded meanings, algorithm name/type, and frame references to source series; include FoR and (if resampled) the registration UID.

## 7.2 FHIR Mapping (Normative Profiles)

TIDeS defines a **minimal working set** of FHIR R4/R5 resources with **StructureDefinitions**.

### 7.2.1 Profile Inventory

* **StructureDefinition/tides-imagingstudy** — binds SeriesInstanceUIDs and DICOM endpoints.
* **StructureDefinition/tides-absorbed-dose** — Observation profile for absorbed dose per ROI (LOINC 89469‑4).
* **StructureDefinition/tides-dose-rate** — Observation (interim) for dose‑rate (UCUM Gy/s).
* **StructureDefinition/tides-diagnosticreport-theranostics** — DiagnosticReport wrapper for the TidesDoseReport.
* **StructureDefinition/tides-provenance** — Provenance with software agent and entity inputs.
* **ValueSet/tides-roi-snomed-radlex** — ROI codes.

### 7.2.2 Cardinalities & Must‑Support

| Element | Cardinality | Must‑Support | Notes |
| --- | --- | --- | --- |
| DiagnosticReport.code | 1..1 | ✓ | fixed to “Theranostics dose report” |
| DiagnosticReport.result | 1..\* | ✓ | Absorbed dose Observations |
| Observation.valueQuantity | 1..1 | ✓ | Gy UCUM |
| Observation.bodySite | 1..1 | ✓ | SNOMED/RadLex |
| Provenance.agent | 1..\* | ✓ | software actor |

### 7.2.3 Example Resources (Canonical)

**DiagnosticReport (Theranostics)**

{  
 "resourceType": "DiagnosticReport",  
 "meta": {"profile": ["http://tides.org/fhir/StructureDefinition/tides-diagnosticreport-theranostics"]},  
 "status": "final",  
 "code": {"coding": [{"system":"http://tides.org/codes","code":"theranostics-dose-report"}]},  
 "subject": {"reference": "Patient/123"},  
 "effectiveDateTime": "2025-09-25T11:03:27Z",  
 "result": [  
 {"reference": "Observation/absdose-liver"},  
 {"reference": "Observation/absdose-kidney"}  
 ],  
 "conclusionCode": [{"coding":[{"system":"http://tides.org/codes/safety","code":"warn","display":"Policy warn"}]}]  
}

**Observation — Absorbed Dose (ROI)**

{  
 "resourceType": "Observation",  
 "id": "absdose-liver",  
 "meta": {"profile": ["http://tides.org/fhir/StructureDefinition/tides-absorbed-dose"]},  
 "status": "final",  
 "code": {"coding": [{"system":"http://loinc.org","code":"89469-4","display":"Absorbed dose"}]},  
 "valueQuantity": {"value": 18.2, "system":"http://unitsofmeasure.org", "code":"Gy"},  
 "bodySite": {"coding":[{"system":"http://snomed.info/sct","code":"74281007","display":"Liver"}]},  
 "derivedFrom": [{"reference":"ImagingStudy/abc"}],  
 "note": [{"text": "uncertainty 9.1% via delta-method"}]  
}

**Observation — Dose‑Rate (ROI)**

{  
 "resourceType": "Observation",  
 "meta": {"profile": ["http://tides.org/fhir/StructureDefinition/tides-dose-rate"]},  
 "status": "final",  
 "code": {"coding": [{"system":"http://tides.org/codes","code":"dose-rate"}]},  
 "valueQuantity": {"value": 8.7, "system":"http://unitsofmeasure.org", "code":"Gy/s"},  
 "bodySite": {"coding":[{"system":"http://snomed.info/sct","code":"74281007","display":"Liver"}]}  
}

**Provenance (software agent)**

{  
 "resourceType": "Provenance",  
 "agent": [{"type": {"text":"software"}, "who": {"display":"tides-cli 1.0.0"}}],  
 "entity": [{"role":"source", "what": {"reference":"Binary/dosemap-rt"}}]  
}

### 7.2.4 StructureDefinition (Excerpt; JSON diff‑style)

{  
 "resourceType": "StructureDefinition",  
 "url": "http://tides.org/fhir/StructureDefinition/tides-absorbed-dose",  
 "type": "Observation",  
 "kind": "resource",  
 "abstract": false,  
 "differential": {  
 "element": [  
 {"id":"Observation.code", "fixedCodeableConcept": {"coding":[{"system":"http://loinc.org","code":"89469-4"}]}},  
 {"id":"Observation.valueQuantity.system", "fixedUri":"http://unitsofmeasure.org"},  
 {"id":"Observation.valueQuantity.code", "fixedCode":"Gy"},  
 {"id":"Observation.bodySite", "mustSupport": true}  
 ]  
 }  
}

### 7.2.5 Bundle Packaging

* Group resources into a **transaction** Bundle for submission; include Binary stubs or DICOMweb Endpoint references for large objects.
* Preserve TIDeS URNs under identifier on each resource.

**Validator rules:** fhir-profile-declared (ERROR), fhir-unit-ucum (ERROR), link-referential (ERROR on broken references).

## 7.3 OMOP Mapping (Analytics‑Ready)

### 7.3.1 DDL (Normative; Extends §2.6.3)

-- Versioning table  
CREATE TABLE tides\_meta (  
 key TEXT PRIMARY KEY,  
 value TEXT NOT NULL  
);  
INSERT INTO tides\_meta(key,value) VALUES ('ddl\_version','1.0.0');  
  
-- Core tables (study, roi dose, provenance)  
CREATE TABLE IF NOT EXISTS tides\_study (  
 tides\_study\_id SERIAL PRIMARY KEY,  
 person\_id INT NOT NULL,  
 study\_uid TEXT NOT NULL UNIQUE,  
 nuclide TEXT, agent TEXT, protocol\_id TEXT,  
 start\_ts TIMESTAMP, end\_ts TIMESTAMP  
);  
  
CREATE TABLE IF NOT EXISTS tides\_roi\_dose (  
 tides\_roi\_dose\_id SERIAL PRIMARY KEY,  
 study\_uid TEXT NOT NULL REFERENCES tides\_study(study\_uid),  
 roi\_code TEXT NOT NULL, roi\_text TEXT,  
 dmean\_gy NUMERIC, dmax\_gy NUMERIC,  
 v10gy NUMERIC, dx\_gy NUMERIC,  
 uncertainty\_pc NUMERIC, uncertainty\_method TEXT,  
 dosemap\_uri TEXT, pk\_model\_id TEXT,  
 created\_at TIMESTAMP DEFAULT now()  
);  
  
CREATE TABLE IF NOT EXISTS tides\_provenance (  
 tides\_prov\_id SERIAL PRIMARY KEY,  
 study\_uid TEXT NOT NULL REFERENCES tides\_study(study\_uid),  
 software TEXT, version TEXT, hash TEXT,  
 inputs JSONB, params JSONB, policy\_packs JSONB  
);  
  
-- Optional table for dose-rate samples (see Ch.4/5)  
CREATE TABLE IF NOT EXISTS tides\_doserate\_samples (  
 id BIGSERIAL PRIMARY KEY,  
 dosemap\_uri TEXT NOT NULL,  
 roi\_code TEXT NOT NULL,  
 t\_s DOUBLE PRECISION NOT NULL,  
 dose\_rate\_gys DOUBLE PRECISION NOT NULL  
);

**Validator rules:** omop-ddl-version (ERROR if missing or version mismatch), unit-ucum (when exporting quantities into numeric fields with implicit units documented).

### 7.3.2 ETL Logic (TIDeS → OMOP)

| TIDeS Field | OMOP Table.Column | Transform |
| --- | --- | --- |
| TidesStudy.id | tides\_study.study\_uid | Verbatim URN |
| context.nuclide | tides\_study.nuclide | Text |
| organSummaries.Dmean\_Gy | tides\_roi\_dose.dmean\_gy | Numeric |
| uncertainty\_pc | tides\_roi\_dose.uncertainty\_pc | Numeric |
| provenance.\* | tides\_provenance.\* | JSONB mirror |

**SQL view for safety (see Ch.6):** tides\_oar\_flags.

## 7.4 Reference Converters (Executable Code)

### 7.4.1 Python — DICOM (RTDOSE/SEG) → TIDeS Extract

import pydicom as dcm  
  
def extract\_rt\_dose(path: str) -> dict:  
 ds = dcm.dcmread(path, force=True)  
 assert ds.DoseUnits.upper() == 'GY'  
 return {  
 'frameOfReferenceUID': ds.ReferencedFrameOfReferenceSequence[0].FrameOfReferenceUID,  
 'grid': {  
 'scaling': float(ds.DoseGridScaling),  
 'offsets': [float(x) for x in ds.GridFrameOffsetVector]  
 },  
 'references': {'images': len(getattr(ds, 'ReferencedImageSequence', []))}  
 }  
  
def extract\_seg(path: str) -> dict:  
 ds = dcm.dcmread(path, force=True)  
 segs = []  
 for s in ds.SegmentSequence:  
 code = s.SegmentedPropertyTypeCodeSequence[0]  
 segs.append({  
 'number': int(s.SegmentNumber),  
 'label': s.SegmentLabel,  
 'code': {'system': 'SNOMED', 'value': code.CodeValue, 'display': code.CodeMeaning},  
 'algorithm': {'type': s.SegmentAlgorithmType, 'name': s.SegmentAlgorithmName}  
 })  
 return {  
 'frameOfReferenceUID': ds.FrameOfReferenceUID,  
 'segments': segs  
 }

### 7.4.2 Python — TIDeS → FHIR Bundle (Minimal)

from typing import List, Dict  
  
def tides\_to\_fhir\_bundle(dosereport: Dict, patient\_ref: str = 'Patient/123') -> Dict:  
 obs = []  
 for s in dosereport['series']:  
 for o in s['organSummaries']:  
 obs.append({  
 'resourceType': 'Observation',  
 'meta': {'profile': ['http://tides.org/fhir/StructureDefinition/tides-absorbed-dose']},  
 'status': 'final',  
 'code': {'coding':[{'system':'http://loinc.org','code':'89469-4','display':'Absorbed dose'}]},  
 'valueQuantity': {'value': o['Dmean\_Gy'], 'system':'http://unitsofmeasure.org','code':'Gy'},  
 'bodySite': {'coding':[{'system':'http://snomed.info/sct','code': o['roiCode']}]}  
 })  
 dr = {  
 'resourceType': 'DiagnosticReport',  
 'meta': {'profile': ['http://tides.org/fhir/StructureDefinition/tides-diagnosticreport-theranostics']},  
 'status': 'final', 'subject': {'reference': patient\_ref},  
 'code': {'coding':[{'system':'http://tides.org/codes','code':'theranostics-dose-report'}]},  
 'result': [{'reference': f'urn:uuid:{i}'} for i,\_ in enumerate(obs)]  
 }  
 bundle = {'resourceType': 'Bundle', 'type': 'collection', 'entry': [{'resource': dr}] + [{'resource': r} for r in obs]}  
 return bundle

### 7.4.3 Node.js — FHIR Validation (Profile Declarations)

function assertProfiles(bundle) {  
 const errs = [];  
 (bundle.entry || []).forEach(e => {  
 const r = e.resource;  
 if (r.resourceType === 'Observation') {  
 const has = r.meta && r.meta.profile && r.meta.profile.includes('http://tides.org/fhir/StructureDefinition/tides-absorbed-dose');  
 if (!has) errs.push({id:'fhir-profile-declared', severity:'ERROR', msg:'Observation profile missing'});  
 const uq = r.valueQuantity;  
 if (!uq || uq.code !== 'Gy' || uq.system !== 'http://unitsofmeasure.org')  
 errs.push({id:'fhir-unit-ucum', severity:'ERROR', msg:'Observation.valueQuantity must be Gy UCUM'});  
 }  
 });  
 return errs;  
}

### 7.4.4 SQL — ETL Upsert from FHIR Bundle

-- Pseudocode using SQL/JSON features  
WITH obs AS (  
 SELECT \* FROM jsonb\_to\_recordset(:bundle::jsonb->'entry') AS e(resource jsonb)  
), dose\_obs AS (  
 SELECT (resource->'resource'->>'resourceType') AS type,  
 resource->'resource' AS r  
 FROM obs WHERE (resource->'resource'->>'resourceType') = 'Observation'  
)  
INSERT INTO tides\_roi\_dose(study\_uid, roi\_code, dmean\_gy)  
SELECT :study\_uid, r->'bodySite'->'coding'->0->>'code', (r->'valueQuantity'->>'value')::numeric  
FROM dose\_obs;

## 7.5 Conformance Testing & CI

* **DICOM round‑trip tests:** Parse tags listed in §7.1.2 from fixtures; assert presence/values; re‑serialize RTDOSE/SEG and compare FoR/units.
* **FHIR validation:** Run HL7 validator against provided StructureDefinitions; enforce UCUM units; verify references.
* **OMOP migration tests:** Apply DDL, run ETL, and assert counts/key constraints.

**Fixtures (build on §11):** 1. case\_registration\_ok — DICOM FoR alignment present; SEG coded; RTDOSE units Gy. 2. case\_ucum\_wrong\_unit\_text — FHIR Observation with non‑UCUM symbol → **FAIL**. 3. case\_legacy\_profile\_C — Minimal organ‑level export; FHIR MAY be partial; OMOP rows present.

## 7.6 Must‑Support Matrix (Profiles A/B/C)

| Capability | A (Clinical‑Full) | B (Research‑Voxel) | C (Legacy‑Organ) |
| --- | --- | --- | --- |
| DICOM FoR/tags present | **MUST** | **MUST** | **SHOULD** |
| SEG coded ROIs | **MUST** | **SHOULD** | **MAY** |
| RTDOSE units/links | **MUST** | **MUST** | **N/A** |
| FHIR DiagnosticReport + Observations | **MUST** | **SHOULD** | **MAY** |
| OMOP DDL populated | **SHOULD** | **SHOULD** | **MAY** |

## 7.7 Validator Rule Set (Interop Layer)

* dicom-required-tags — required DICOM header elements exist (ERROR)
* seg-coded-meanings — coded ROI meanings in SEG (ERROR A/B)
* rtdose-for-match — RTDOSE aligns to FoR and declares units Gy (ERROR)
* parametric-rwvm — Parametric maps include RWVM with UCUM (ERROR)
* fhir-profile-declared — FHIR profiles on resources (ERROR)
* fhir-unit-ucum — UCUM codes in valueQuantity (ERROR)
* omop-ddl-version — OMOP schema version present (ERROR)
* link-referential — FHIR/DICOM references resolvable (ERROR)

## 7.8 TIDeS‑CHK‑7 — Interop Readiness Checklist

* DICOM: Required tags present; FoR consistent; SEG coded; RTDOSE units Gy.
* FHIR: DiagnosticReport + ROI Observations with profiles; UCUM units; references valid.
* OMOP: DDL installed; ETL populated; policy/safety views validated.
* Provenance: software/version/hash and policy packs included in all exports.
* CI: round‑trip fixtures pass; validator badges achieved for target profile.

## 7.9 FAQs

**Q: Can we omit FHIR if we only use OMOP?**  
A: Yes, but for clinical exchange Profile A expects FHIR; Profile B may accept OMOP‑only research pipelines.

**Q: How do we reference large DICOM payloads in FHIR?**  
A: Use Endpoint/DICOMweb references or Binary stubs with hashes; keep TIDeS URNs in identifier.

**Q: What about R5 vs R4?**  
A: Profiles are defined for R4; R5 snapshots are provided as informative. Do not mix without explicit profile URLs.

## 7.10 Chapter Summary

* DICOM encodes **where** and **how** pixels and dose live; FHIR carries **clinical facts**; OMOP stores **analytics facts**. TIDeS binds them with precise, executable mappings and profiles.
* With the tag lists, profile diffs, DDL, and reference code herein, interoperability is **repeatable, testable, and portable** across vendors and sites.

**End of Chapter 7 (Normative).**

# TIDeS Handbook — Chapter 8

## Schemas (Normative JSON Schema 2020-12 + Codegen)

**Purpose.** Provide the **authoritative, executable** schemas for all TIDeS resources using **JSON Schema 2020‑12**, plus reference validators (Python/Node), type generation (TypeScript/Go), and CI harness. These schemas are the normative contract for TIDeS bundles and artifacts.

**Audience.** Implementers, vendors, integrators, and registry/validator developers.

**Outcome.** You will (a) validate TIDeS bundles deterministically, (b) generate types from the same source of truth, (c) embed $id/$anchor for cross‑refs, (d) ensure SemVer discipline across /schemas, and (e) wire schemas to the Chapter 9 validator.

**Normative keywords:** **MUST**, **SHOULD**, **MAY**.

## 8.0 Schema Philosophy & Packaging (Normative)

1. **Single Source of Truth.** All resource contracts live under /schemas/1.0.0/ with one file per resource; shared definitions in $defs. Every file **MUST** have $schema, $id, title, and version.
2. **SemVer.** The schema directory mirrors spec SemVer. A patch release **MUST NOT** break validation for previously conformant instances.
3. **Cross‑refs.** Use absolute $id URLs and $ref across files; provide local $anchor for intra‑file targets.
4. **Profiles.** Profile A/B/C constraints are implemented as **separate, additive schemas** that allOf the base schemas.
5. **Formats.** Use standard formats (e.g., date-time) and register custom formats (UCUM token, URN) in validators.

Directory layout:

/schemas/1.0.0/  
 tides-bundle.schema.json  
 tides-study.schema.json  
 tides-imaging.schema.json  
 tides-seg.schema.json  
 tides-calibration.schema.json  
 tides-pkmodel.schema.json  
 tides-dosemap.schema.json  
 tides-dosereport.schema.json  
 tides-outcome.schema.json  
 tides-timings.schema.json  
 tides-kernel.schema.json  
 tides-policy.schema.json # from Chapter 6  
 profiles/  
 profile-A.schema.json  
 profile-B.schema.json  
 profile-C.schema.json  
 $defs/  
 common.json  
 ucum.json  
 identifiers.json

## 8.1 Common Definitions ($defs) (Normative)

**$defs/common.json**

{  
 "$id": "https://tides.org/schemas/1.0.0/$defs/common.json",  
 "$schema": "https://json-schema.org/draft/2020-12/schema",  
 "title": "TIDeS Common",  
 "type": "object",  
 "$defs": {  
 "SpecVersion": { "type": "string", "const": "1.0.0" },  
 "URN": {  
 "type": "string",  
 "pattern": "^urn:tides:[a-z]+(?::[A-Za-z0-9.\_-]+)?:[0-9a-fA-F-]{36}$",  
 "$comment": "urn:tides:<type>[:<flavor>]:<UUID>"  
 },  
 "UCUMGy": { "type": "string", "const": "Gy" },  
 "UCUMGyPerS": { "type": "string", "const": "Gy/s" },  
 "UCUMOne": { "type": "string", "const": "1" },  
 "DateTime": { "type": "string", "format": "date-time" },  
 "Codeable": {  
 "type": "object",  
 "required": ["code","system"],  
 "properties": {  
 "code": {"type": "string"},  
 "system": {"type": "string", "enum": ["SNOMED","RadLex","LOCAL"]},  
 "display": {"type": "string"}  
 }  
 },  
 "Provenance": {  
 "type": "object",  
 "required": ["software","version","hash","inputs"],  
 "properties": {  
 "software": {"type": "string"},  
 "version": {"type": "string"},  
 "hash": {"type": "string"},  
 "inputs": {"type": "array", "items": {"type": "string"}},  
 "params": {"type": "object"},  
 "policyPacks": {"type": "array", "items": {"type": "object", "required": ["name","version","specMajor"], "properties": {"name": {"type":"string"}, "version": {"type": "string"}, "specMajor": {"type":"integer"}}}}  
 }  
 },  
 "FoR": { "type": "string", "pattern": "^(?:[0-9]+)(?:\\.[0-9]+)+$" },  
 "Grid": {  
 "type": "object",  
 "required": ["spacing\_mm","size"],  
 "properties": {  
 "spacing\_mm": {"type": "array", "items": {"type": "number"}, "minItems": 3, "maxItems": 3},  
 "size": {"type": "array", "items": {"type": "integer"}, "minItems": 3, "maxItems": 3},  
 "offsets\_mm": {"type": "array", "items": {"type": "number"}, "minItems": 3, "maxItems": 3}  
 }  
 }  
 }  
}

**$defs/identifiers.json**

{  
 "$id": "https://tides.org/schemas/1.0.0/$defs/identifiers.json",  
 "$schema": "https://json-schema.org/draft/2020-12/schema",  
 "$defs": {  
 "StudyURN": { "type": "string", "pattern": "^urn:tides:study:[0-9a-fA-F-]{36}$" },  
 "SubjectURN": { "type": "string", "pattern": "^urn:tides:subject:[0-9a-fA-F-]{36}$" },  
 "ImagingURN": { "type": "string", "pattern": "^urn:tides:imaging:[0-9a-fA-F-]{36}$" },  
 "SegURN": { "type": "string", "pattern": "^urn:tides:seg:[0-9a-fA-F-]{36}$" },  
 "CalibURN": { "type": "string", "pattern": "^urn:tides:calibration:[0-9a-fA-F-]{36}$" },  
 "PKURN": { "type": "string", "pattern": "^urn:tides:pkmodel(?::[A-Za-z0-9.\_-]+)?:[0-9a-fA-F-]{36}$" },  
 "DoseMapURN": { "type": "string", "pattern": "^urn:tides:dosemap(?::[A-Za-z0-9.\_-]+)?:[0-9a-fA-F-]{36}$" },  
 "DoseReportURN": { "type": "string", "pattern": "^urn:tides:dosereport:[0-9a-fA-F-]{36}$" },  
 "OutcomeURN": { "type": "string", "pattern": "^urn:tides:outcome:[0-9a-fA-F-]{36}$" }  
 }  
}

## 8.2 Timing Schema (Authoritative) — tides-timings.schema.json

{  
 "$id": "https://tides.org/schemas/1.0.0/tides-timings.schema.json",  
 "$schema": "https://json-schema.org/draft/2020-12/schema",  
 "title": "TIDeS Timing",  
 "type": "object",  
 "required": ["injectionStart","frames"],  
 "properties": {  
 "injectionStart": {"$ref": "https://tides.org/schemas/1.0.0/$defs/common.json#/$defs/DateTime"},  
 "frames": {  
 "type": "array",  
 "minItems": 1,  
 "items": {  
 "type": "object",  
 "required": ["tMid\_s","duration\_s"],  
 "properties": {  
 "tMid\_s": {"type": "number", "minimum": 0},  
 "duration\_s": {"type": "number", "exclusiveMinimum": 0}  
 }  
 }  
 },  
 "fitWindow\_s": {"type": "array", "minItems": 2, "maxItems": 2, "items": {"type": "number", "minimum": 0}},  
 "rawFrames": {"type": "array"},  
 "clockSync": {  
 "type": "object",  
 "properties": {  
 "scannerMinusIS\_ms": {"type": "number"},  
 "applied": {"type": "boolean"},  
 "residualClockSkew\_ms": {"type": "number"}  
 }  
 }  
 }  
}

## 8.3 Study & Imaging — tides-study.schema.json, tides-imaging.schema.json

**Study**

{  
 "$id": "https://tides.org/schemas/1.0.0/tides-study.schema.json",  
 "$schema": "https://json-schema.org/draft/2020-12/schema",  
 "title": "TidesStudy",  
 "type": "object",  
 "required": ["id","specVersion","subject"],  
 "properties": {  
 "id": {"$ref": "https://tides.org/schemas/1.0.0/$defs/identifiers.json#/$defs/StudyURN"},  
 "specVersion": {"$ref": "https://tides.org/schemas/1.0.0/$defs/common.json#/$defs/SpecVersion"},  
 "subject": {"$ref": "https://tides.org/schemas/1.0.0/$defs/identifiers.json#/$defs/SubjectURN"},  
 "label": {"type": "string"},  
 "context": {"type": "object"},  
 "timeframe": {"type": "object", "properties": {"start": {"type":"string","format":"date-time"}, "end": {"type":"string","format":"date-time"}}},  
 "policyPacks": {"type": "array", "items": {"type": "object", "required": ["name","version","specMajor"], "properties": {"name":{"type":"string"},"version":{"type":"string"},"specMajor":{"type":"integer"}}}},  
 "attachments": {"type": "array", "items": {"type":"object","properties":{"uri":{"type":"string"},"type":{"type":"string"},"description":{"type":"string"},"hash":{"type":"string"}}}},  
 "provenance": {"$ref": "https://tides.org/schemas/1.0.0/$defs/common.json#/$defs/Provenance"}  
 }  
}

**Imaging**

{  
 "$id": "https://tides.org/schemas/1.0.0/tides-imaging.schema.json",  
 "$schema": "https://json-schema.org/draft/2020-12/schema",  
 "title": "TidesImaging",  
 "type": "object",  
 "required": ["id","frameOfReferenceUID","timings","dicomSeries"],  
 "properties": {  
 "id": {"$ref": "https://tides.org/schemas/1.0.0/$defs/identifiers.json#/$defs/ImagingURN"},  
 "frameOfReferenceUID": {"$ref": "https://tides.org/schemas/1.0.0/$defs/common.json#/$defs/FoR"},  
 "dicomSeries": {"type": "array", "items": {"type": "string"}},  
 "timings": {"$ref": "https://tides.org/schemas/1.0.0/tides-timings.schema.json"},  
 "realWorldValueMaps": {"type": "array", "items": {"type": "object"}},  
 "calibration": {"$ref": "https://tides.org/schemas/1.0.0/$defs/identifiers.json#/$defs/CalibURN"},  
 "seg": {"type": "array", "items": {"$ref": "https://tides.org/schemas/1.0.0/$defs/identifiers.json#/$defs/SegURN"}},  
 "provenance": {"$ref": "https://tides.org/schemas/1.0.0/$defs/common.json#/$defs/Provenance"}  
 }  
}

## 8.4 Segmentation & Calibration — tides-seg.schema.json, tides-calibration.schema.json

**Seg**

{  
 "$id": "https://tides.org/schemas/1.0.0/tides-seg.schema.json",  
 "$schema": "https://json-schema.org/draft/2020-12/schema",  
 "title": "TidesSeg",  
 "type": "object",  
 "required": ["id","frameOfReferenceUID","roi"],  
 "properties": {  
 "id": {"$ref": "https://tides.org/schemas/1.0.0/$defs/identifiers.json#/$defs/SegURN"},  
 "frameOfReferenceUID": {"$ref": "https://tides.org/schemas/1.0.0/$defs/common.json#/$defs/FoR"},  
 "roi": {  
 "type": "array",  
 "minItems": 1,  
 "items": {  
 "type": "object",  
 "required": ["code","codeSystem","text"],  
 "properties": {  
 "code": {"type": "string"},  
 "codeSystem": {"type": "string", "enum": ["SNOMED","RadLex","LOCAL"]},  
 "text": {"type": "string"},  
 "algorithm": {"type": "object", "properties": {"type": {"type":"string"}, "name": {"type":"string"}}}  
 }  
 }  
 },  
 "provenance": {"$ref": "https://tides.org/schemas/1.0.0/$defs/common.json#/$defs/Provenance"}  
 }  
}

**Calibration**

{  
 "$id": "https://tides.org/schemas/1.0.0/tides-calibration.schema.json",  
 "$schema": "https://json-schema.org/draft/2020-12/schema",  
 "title": "TidesCalibration",  
 "type": "object",  
 "required": ["id","scannerFactor\_Bq\_per\_count","decayCorrectionPolicy"],  
 "properties": {  
 "id": {"$ref": "https://tides.org/schemas/1.0.0/$defs/identifiers.json#/$defs/CalibURN"},  
 "scannerFactor\_Bq\_per\_count": {"type": "number", "exclusiveMinimum": 0},  
 "decayCorrectionPolicy": {"type": "string", "enum": ["acq-start","frame-mid","injection-start","none"]},  
 "crossCal": {"type": "object", "properties": {"doseCalibratorId": {"type":"string"}, "factor": {"type":"number"}, "date": {"type":"string","format":"date"}}},  
 "provenance": {"$ref": "https://tides.org/schemas/1.0.0/$defs/common.json#/$defs/Provenance"}  
 }  
}

## 8.5 PK Model — tides-pkmodel.schema.json

{  
 "$id": "https://tides.org/schemas/1.0.0/tides-pkmodel.schema.json",  
 "$schema": "https://json-schema.org/draft/2020-12/schema",  
 "title": "TidesPKModel",  
 "type": "object",  
 "required": ["id","scope","fit"],  
 "properties": {  
 "id": {"$ref": "https://tides.org/schemas/1.0.0/$defs/identifiers.json#/$defs/PKURN"},  
 "scope": {"type": "string", "enum": ["voxel","roi"]},  
 "priors": {"type": "object"},  
 "fit": {  
 "type": "object",  
 "required": ["method","params"],  
 "properties": {  
 "method": {"type": "string", "enum": ["WLS","NLS","Bayes"]},  
 "params": {"type": "object"},  
 "covariance": {"type": "array", "items": {"type": "array", "items": {"type": "number"}}},  
 "diagnostics": {"type": "object", "properties": {"R2": {"type":"number"}, "AIC": {"type":"number"}, "BIC": {"type":"number"}}}  
 }  
 },  
 "tia\_Bq\_s": {"type": "number"},  
 "tail\_method": {"type": "string"},  
 "provenance": {"$ref": "https://tides.org/schemas/1.0.0/$defs/common.json#/$defs/Provenance"}  
 }  
}

## 8.6 Dose Map & Kernel — tides-dosemap.schema.json, tides-kernel.schema.json

**Kernel**

{  
 "$id": "https://tides.org/schemas/1.0.0/tides-kernel.schema.json",  
 "$schema": "https://json-schema.org/draft/2020-12/schema",  
 "title": "Kernel Metadata",  
 "type": "object",  
 "required": ["formalism","nuclide","medium","grid\_mm","version"],  
 "properties": {  
 "formalism": {"type": "string", "enum": ["S-value","MC"]},  
 "nuclide": {"type": "string"},  
 "medium": {"type": "string"},  
 "grid\_mm": {"type": "array", "items": {"type":"number"}, "minItems": 3, "maxItems": 3},  
 "support\_vox": {"type": "integer", "minimum": 1},  
 "version": {"type": "string"},  
 "sourceURI": {"type": "string"}  
 }  
}

**DoseMap**

{  
 "$id": "https://tides.org/schemas/1.0.0/tides-dosemap.schema.json",  
 "$schema": "https://json-schema.org/draft/2020-12/schema",  
 "title": "TidesDoseMap",  
 "type": "object",  
 "required": ["id","frameOfReferenceUID","kernel","source"],  
 "properties": {  
 "id": {"$ref": "https://tides.org/schemas/1.0.0/$defs/identifiers.json#/$defs/DoseMapURN"},  
 "frameOfReferenceUID": {"$ref": "https://tides.org/schemas/1.0.0/$defs/common.json#/$defs/FoR"},  
 "rtDose": {"type": "string"},  
 "kernel": {"$ref": "https://tides.org/schemas/1.0.0/tides-kernel.schema.json"},  
 "source": {"$ref": "https://tides.org/schemas/1.0.0/$defs/identifiers.json#/$defs/PKURN"},  
 "flashCoverage\_pc": {"type": "number", "minimum": 0, "maximum": 100},  
 "attachments": {"type": "array", "items": {"type": "object"}},  
 "provenance": {"$ref": "https://tides.org/schemas/1.0.0/$defs/common.json#/$defs/Provenance"}  
 }  
}

## 8.7 Dose Report — tides-dosereport.schema.json (Expanded)

{  
 "$id": "https://tides.org/schemas/1.0.0/tides-dosereport.schema.json",  
 "$schema": "https://json-schema.org/draft/2020-12/schema",  
 "title": "TIDeS Dose Report",  
 "type": "object",  
 "required": ["id","specVersion","subject","series"],  
 "properties": {  
 "id": {"$ref": "https://tides.org/schemas/1.0.0/$defs/identifiers.json#/$defs/DoseReportURN"},  
 "specVersion": {"$ref": "https://tides.org/schemas/1.0.0/$defs/common.json#/$defs/SpecVersion"},  
 "subject": {"$ref": "https://tides.org/schemas/1.0.0/$defs/identifiers.json#/$defs/SubjectURN"},  
 "series": {  
 "type": "array",  
 "items": {  
 "type": "object",  
 "required": ["doseMap","organSummaries"],  
 "properties": {  
 "doseMap": {"$ref": "https://tides.org/schemas/1.0.0/$defs/identifiers.json#/$defs/DoseMapURN"},  
 "organSummaries": {  
 "type": "array",  
 "items": {  
 "type": "object",  
 "required": ["roiCode","Dmean\_Gy","uncertainty\_pc","uncertaintyMethod"],  
 "properties": {  
 "roiCode": {"type": "string"},  
 "roiCodeText": {"type": "string"},  
 "Dmean\_Gy": {"type": "number"},  
 "Dmax\_Gy": {"type": "number"},  
 "Dx\_Gy": {"type": "number"},  
 "Vx": {"type": "object", "additionalProperties": {"type": "number"}},  
 "uncertainty\_pc": {"type": "number", "minimum": 0},  
 "uncertaintyMethod": {"type": "string", "enum": ["delta-method","bootstrap","mixed"]}  
 }  
 }  
 }  
 }  
 }  
 },  
 "policyResults": {"type": "object"},  
 "provenance": {"$ref": "https://tides.org/schemas/1.0.0/$defs/common.json#/$defs/Provenance"}  
 }  
}

## 8.8 Outcome — tides-outcome.schema.json

{  
 "$id": "https://tides.org/schemas/1.0.0/tides-outcome.schema.json",  
 "$schema": "https://json-schema.org/draft/2020-12/schema",  
 "title": "TidesOutcome",  
 "type": "object",  
 "required": ["id","timepoint"],  
 "properties": {  
 "id": {"$ref": "https://tides.org/schemas/1.0.0/$defs/identifiers.json#/$defs/OutcomeURN"},  
 "timepoint": {"type": "string", "format": "date-time"},  
 "response": {"type": "string"},  
 "toxicity": {"type": "array", "items": {"type":"object", "properties": {"code": {"type":"string"}, "grade": {"type":"integer","minimum": 1, "maximum": 5}, "onsetDate": {"type":"string","format":"date"}}}},  
 "links": {"type": "array", "items": {"type": "string"}},  
 "provenance": {"$ref": "https://tides.org/schemas/1.0.0/$defs/common.json#/$defs/Provenance"}  
 }  
}

## 8.9 Bundle — tides-bundle.schema.json (Top‑Level)

{  
 "$id": "https://tides.org/schemas/1.0.0/tides-bundle.schema.json",  
 "$schema": "https://json-schema.org/draft/2020-12/schema",  
 "title": "TIDeS Bundle",  
 "type": "object",  
 "required": ["specVersion","id","study","resources"],  
 "properties": {  
 "specVersion": {"$ref": "https://tides.org/schemas/1.0.0/$defs/common.json#/$defs/SpecVersion"},  
 "id": {"type": "string", "pattern": "^urn:tides:bundle:[0-9a-fA-F-]{3,}$"},  
 "study": {"$ref": "https://tides.org/schemas/1.0.0/$defs/identifiers.json#/$defs/StudyURN"},  
 "resources": {"type": "array", "minItems": 1},  
 "provenance": {"$ref": "https://tides.org/schemas/1.0.0/$defs/common.json#/$defs/Provenance"}  
 },  
 "$comment": "Resource items must also pass their specific schemas; enforced by the validator (Chapter 9)."  
}

## 8.10 Profiles — profiles/profile-\*.schema.json

**Profile A (extract)**

{  
 "$id": "https://tides.org/schemas/1.0.0/profiles/profile-A.schema.json",  
 "$schema": "https://json-schema.org/draft/2020-12/schema",  
 "title": "TIDeS Profile A",  
 "allOf": [  
 { "$ref": "https://tides.org/schemas/1.0.0/tides-bundle.schema.json" }  
 ],  
 "properties": {  
 "resources": {  
 "type": "array",  
 "contains": {"type":"object","properties": {"resourceType": {"const":"TidesDoseMap"}}},  
 "minItems": 1  
 }  
 }  
}

(Profiles B/C similar with differing contains/required sets.)

## 8.11 Reference Validators (Executable)

### 8.11.1 Python (jsonschema)

import json, sys  
from jsonschema import Draft202012Validator, RefResolver  
  
SCHEMA\_BASE = "https://tides.org/schemas/1.0.0/"  
  
# Preload schemas into a store (in production serve over HTTPS)  
from pathlib import Path  
store = {}  
for p in Path('schemas/1.0.0').rglob('\*.schema.json'):  
 doc = json.loads(Path(p).read\_text())  
 store[doc['$id']] = doc  
  
resolver = RefResolver.from\_schema(store[SCHEMA\_BASE+'tides-bundle.schema.json'], store=store)  
validator = Draft202012Validator(store[SCHEMA\_BASE+'tides-bundle.schema.json'], resolver=resolver)  
  
inst = json.loads(sys.stdin.read())  
errors = sorted(validator.iter\_errors(inst), key=lambda e: e.path)  
for e in errors:  
 print(f"ERROR at {'/'.join(map(str,e.path))}: {e.message}")  
sys.exit(1 if errors else 0)

### 8.11.2 Node.js (ajv)

const Ajv = require('ajv/dist/2020');  
const addFormats = require('ajv-formats');  
const fs = require('fs');  
const path = require('path');  
  
const base = path.join(\_\_dirname, 'schemas/1.0.0');  
const ajv = new Ajv({ strict: true, allErrors: true });  
addFormats(ajv);  
  
function load(file) { return JSON.parse(fs.readFileSync(path.join(base, file))); }  
const files = fs.readdirSync(base).filter(f => f.endsWith('.schema.json'));  
const schemas = files.map(load);  
schemas.forEach(s => ajv.addSchema(s, s.$id));  
const validate = ajv.getSchema('https://tides.org/schemas/1.0.0/tides-bundle.schema.json');  
  
const data = JSON.parse(fs.readFileSync(process.argv[2]));  
if (!validate(data)) {  
 console.error(validate.errors);  
 process.exit(1);  
}  
console.log('OK');

### 8.11.3 Custom Formats (URN, UCUM) — Node

ajv.addFormat('tides-urn', /^urn:tides:[a-z]+(?::[A-Za-z0-9.\_-]+)?:[0-9a-fA-F-]{36}$/);  
ajv.addFormat('ucum-gy', /^Gy$/);  
ajv.addFormat('ucum-gys', /^Gy\/s$/);

## 8.12 Code Generation (Types)

### 8.12.1 TypeScript (quicktype/openapi‑typescript‑validator style)

$ quicktype --src-lang schema --lang ts --top-level TidesBundle \  
 schemas/1.0.0/tides-bundle.schema.json -o types/tides.ts

**Excerpt (types/tides.ts):**

export interface TidesBundle {  
 specVersion: '1.0.0';  
 id: string;  
 study: string;  
 resources: any[];  
}

### 8.12.2 Go (quicktype‑go)

$ quicktype --src-lang schema --lang go --top-level TidesBundle \  
 schemas/1.0.0/tides-bundle.schema.json -o types/tides.go

## 8.13 CI Wiring (GitHub Actions)

**.github/workflows/validate.yml**

name: TIDeS Schema & Fixtures  
on: [push, pull\_request]  
jobs:  
 validate:  
 runs-on: ubuntu-latest  
 steps:  
 - uses: actions/checkout@v4  
 - uses: actions/setup-node@v4  
 with: { node-version: '20' }  
 - run: npm i ajv ajv-formats  
 - name: Validate fixtures  
 run: |  
 node validator/ajv-validate.js examples/case\_pass\_minimal/bundle.json || exit 1  
 node validator/ajv-validate.js examples/case\_registration\_ok/bundle.json || exit 1

## 8.14 Worked Example (Bundle Pass)

{  
 "specVersion": "1.0.0",  
 "id": "urn:tides:bundle:fc1f...",  
 "study": "urn:tides:study:1b2f...",  
 "resources": [  
 {"resourceType":"TidesStudy","id":"urn:tides:study:1b2f...","subject":"urn:tides:subject:9a2f..."},  
 {"resourceType":"TidesImaging","id":"urn:tides:imaging:aa11...","frameOfReferenceUID":"1.2.826...","timings":{"injectionStart":"2025-09-25T11:03:27Z","frames":[{"tMid\_s":600,"duration\_s":120}]}},  
 {"resourceType":"TidesDoseMap","id":"urn:tides:dosemap:177Lu:dd44...","frameOfReferenceUID":"1.2.826...","kernel":{"formalism":"S-value","nuclide":"177Lu","medium":"ICRP-soft","grid\_mm":[2,2,2],"version":"v1"},"source":"urn:tides:pkmodel:monoexp:cc33..."},  
 {"resourceType":"TidesDoseReport","id":"urn:tides:dosereport:ee55...","subject":"urn:tides:subject:9a2f...","series":[{"doseMap":"urn:tides:dosemap:177Lu:dd44...","organSummaries":[{"roiCode":"74281007","roiCodeText":"liver","Dmean\_Gy":18.2,"Dmax\_Gy":21.5,"uncertainty\_pc":9.1,"uncertaintyMethod":"delta-method"}]}]}  
 ],  
 "provenance": {"software":"tides-cli","version":"1.0.0","hash":"<sha>","inputs":["dicom:..."],"policyPacks":[{"name":"policy-oar","version":"1.0.0","specMajor":1}]}  
}

## 8.15 Traceability & Rule Mapping (Schema ↔ Validator)

* id patterns → validator id-namespace, id-uuid.
* frameOfReferenceUID present → validator registration-fuid.
* specVersion const → validator spec-version.
* UCUM unit enums → validator unit-ucum, unit-dose-gy, unit-doserate.

## 8.16 TIDeS‑CHK‑8 — Schema Readiness Checklist

* $id/$schema present in every file; SemVer aligned.
* $defs reusable; cross‑refs resolve; anchors documented.
* Profile overlays compile and validate fixtures.
* Node/Python validators pass all canonical cases.
* CI workflow executes schema checks on PR.

## 8.17 FAQs

**Q: Why JSON Schema and not XSD/Protobuf?**  
A: JSON Schema 2020‑12 is web‑native, widely tooled, and supports $id/$ref with strong validation and codegen pathways; it maps cleanly to FHIR JSON.

**Q: How do we evolve without breaking?**  
A: Use SemVer. Add new optional fields in minors; breaking changes only in majors with migration notes and a parallel directory (e.g., /schemas/2.0.0/).

**Q: Can we bundle all schemas into one file?**  
A: Yes for offline validators—publish an aggregated tides-all.schema.json built in CI. $ids must remain unique.

## 8.18 Chapter Summary

* Schemas are **normative contracts**: precise, versioned, and cross‑referenced.
* Reference validators and CI make them executable.
* Type generation lets implementers consume TIDeS safely in any language.

**End of Chapter 8 (Normative).**

# TIDeS Handbook — Chapter 9

## Validator Architecture (Executable, Normative)

**Purpose.** Define the **reference validator** that enforces TIDeS conformance. This chapter specifies the system design, rule evaluation model, profiles, APIs/CLIs, report formats, plug‑ins, performance, security, CI wiring, and full reference code so independent sites get **identical validation results** from the same bundle.

**Audience.** Pipeline engineers, QA leads, vendors, auditors, regulators.

**Outcome.** You will: (a) run the validator in CLI or HTTP mode, (b) integrate rule packs and profiles, (c) emit JSON/Text/HTML/SARIF reports, (d) extend safely with plug‑ins, (e) wire CI to badges, and (f) guarantee determinism.

**Normative keywords:** **MUST**, **SHOULD**, **MAY**.

## 9.0 Principles & Scope (Normative)

1. **Deterministic.** Same inputs → same outputs (including ordering). Randomness **MUST NOT** influence outcomes.
2. **Pure & Offline.** Validation never exfiltrates PHI; web calls are disabled by default. External lookups **MUST** be opt‑in.
3. **Layered.** The validator has layers: *Syntax* (JSON Schema), *Semantics* (rule pack), *Interop* (DICOM/FHIR/OMOP), *Profiles* (A/B/C), *Policy* (Chapter 6).
4. **Traceable.** Every rule maps to a spec clause (trace map) and to fixtures (§11). Reports contain clickable anchors.
5. **Composable.** Plug‑ins for DICOM parsers, FHIR validation, and custom metrics are safely sandboxed.
6. **Stability & Versioning.** The validator observes SemVer; rule IDs are stable across patch releases.

## 9.1 System Overview

+---------------------------+ +-------------------+  
| Input Bundle (JSON) | ---> | Syntax Validator |-- JSON Schema 2020-12  
+---------------------------+ +-------------------+  
 |  
 v  
 +----------------+  
 | Rule Engine |-- Rule Pack (JSON)  
 | (Profiles) |  
 +----------------+  
 |  
 +----------+----------+  
 | |  
 v v  
 +---------------+ +---------------+  
 | Interop Checks| | Policy Engine |  
 | (DICOM/FHIR) | | (Chapter 6) |  
 +---------------+ +---------------+  
 | |  
 +----------+----------+  
 |  
 v  
 +-----------------+  
 | Report Writers |  
 | JSON/TXT/HTML |  
 +-----------------+

## 9.2 Data Model & Inputs

* **Primary input**: A TIDeS Bundle JSON that references embedded or external artifacts (RTDOSE, SEG, FHIR, etc.).
* **Aux inputs**: (a) **Rule Pack** (validator/rules/expanded.json), (b) **Policy Packs** (Chapter 6), (c) **Schemas** (Chapter 8), (d) **Profiles** (profiles/A|B|C).
* **CLI flags** override pack paths.

**Normative:** The validator **MUST** fail fast if the schema $id or specVersion mismatches the rule pack specMajor/minor constraints.

## 9.3 Rule Model (Normative)

A **rule** has: id, severity (INFO|WARN|ERROR|BLOCK), desc, when (JSONPath/expr), test (predicate), onFail (message template), profile filters, links (traceability).

### 9.3.1 Rule Pack (Authoritative JSON)

{  
 "$id": "https://tides.org/validator/1.0.0/rules/expanded.json",  
 "version": "1.0.0",  
 "specVersion": "1.0.0",  
 "rules": [  
 {"id":"spec-version","severity":"ERROR","desc":"Bundle must include specVersion=1.0.0","when":"$","test":"@.specVersion=='1.0.0'","links":["§0","§8.9"],"profiles":["A","B","C"]},  
 {"id":"timing-injection","severity":"ERROR","desc":"Must include injectionStart","when":"$.resources[\*].timings","test":"@ && @.injectionStart","links":["§4.0"],"profiles":["A","B","C"]},  
 {"id":"unit-ucum","severity":"ERROR","desc":"Quantities must use UCUM","when":"$..valueQuantity","test":"@.system=='http://unitsofmeasure.org'","links":["§1.1","§7.2"],"profiles":["A","B"]},  
 {"id":"registration-fuid","severity":"ERROR","desc":"doseMap must include FoR","when":"$..[?(@.resourceType=='TidesDoseMap')]","test":"@.frameOfReferenceUID","links":["§3.1","§7.1"],"profiles":["A","B"]},  
 {"id":"provenance","severity":"ERROR","desc":"Provenance must include software+hash","when":"$.provenance","test":"@.software && @.hash","links":["§1.7","§8.1"],"profiles":["A","B","C"]},  
 {"id":"constraint-fail","severity":"ERROR","desc":"Organ dose exceeds policy limit","when":"$.policyResults.findings[\*]","test":"@.decision!='ERROR'","negate":true,"links":["§6.1-6.3"],"profiles":["A"]}  
 ]  
}

**Normative:** id strings are globally unique and **MUST NOT** be recycled.

## 9.4 Evaluation Engine

1. **Load** schema store and validate syntax.
2. **Load** rule pack + profile overlay; resolve profiles filter.
3. **Build** a deterministic JSON node index (preorder traversal) for stable error ordering.
4. **Evaluate** when selectors (JSONPath 2.0 subset) to produce candidate nodes.
5. **Apply** test predicates (JSONLogic or safe CEL) per node.
6. **Collect** failures/successes; attach trace info (pointer, links).
7. **Aggregate** severity → PASS|WARN|ERROR|BLOCK overall and per profile.

**Determinism:** Node iteration order is by canonical JSON pointer sort; floating‑point comparisons use ulp‑bounded tolerances where relevant (units normalized beforehand).

## 9.5 Profiles (A/B/C) (Normative)

* **A (Clinical‑Full)**: voxel RTDOSE/SEG/DICOM required; FHIR DiagnosticReport required; safety policy enforced.
* **B (Research‑Voxel)**: voxel dose and FoR required; FHIR recommended; safety policy recommended.
* **C (Legacy‑Organ)**: organ summaries only; timing/units/provenance still enforced.

**Selection:** CLI --profile A|B|C.

**Badge:** A-PASS, B-PASS, C-PASS issued when all **MUST** rules pass for the selected profile.

## 9.6 Reference Implementation — Python

### 9.6.1 Package Layout

validator/  
 \_\_init\_\_.py  
 cli.py  
 engine.py  
 jsonpath.py  
 jsonlogic\_safe.py  
 reporters/  
 json.py  
 text.py  
 html.py  
 sarif.py  
 io/  
 dicom.py  
 fhir.py  
 omop.py  
 rules/  
 expanded.json  
 policy/  
 policy-oar-1.0.0.yaml

### 9.6.2 Engine Core (engine.py)

from \_\_future\_\_ import annotations  
from dataclasses import dataclass  
from typing import Any, Dict, List, Iterable  
import json  
  
@dataclass  
class Rule:  
 id: str  
 severity: str  
 desc: str  
 when: str  
 test: str | None = None  
 negate: bool = False  
 profiles: List[str] = None  
 links: List[str] = None  
  
@dataclass  
class Finding:  
 ruleId: str  
 severity: str  
 pointer: str  
 message: str  
 links: List[str]  
  
class Engine:  
 def \_\_init\_\_(self, rules: List[Rule], profile: str):  
 self.rules = [r for r in rules if (not r.profiles) or profile in r.profiles]  
 self.profile = profile  
  
 def evaluate(self, inst: Dict[str, Any]) -> List[Finding]:  
 from .jsonpath import select  
 from .jsonlogic\_safe import test\_expr  
 findings: List[Finding] = []  
 for r in self.rules:  
 nodes = select(inst, r.when) # list of (pointer, node)  
 for ptr, node in nodes:  
 ok = True  
 if r.test is not None:  
 ok = test\_expr(r.test, node)  
 if r.negate:  
 ok = not ok  
 if not ok:  
 findings.append(Finding(r.id, r.severity, ptr, r.desc, r.links or []))  
 return sorted(findings, key=lambda f: (f.severity, f.ruleId, f.pointer))

### 9.6.3 Safe JSONLogic (jsonlogic\_safe.py)

import json, math  
  
ALLOWED\_FUNCS = {  
 '==': lambda a,b: a==b,  
 '!=': lambda a,b: a!=b,  
 '>': lambda a,b: a>b,  
 '>=': lambda a,b: a>=b,  
 '<': lambda a,b: a<b,  
 '<=': lambda a,b: a<=b,  
 'and': lambda \*args: all(args),  
 'or': lambda \*args: any(args),  
 '!!': lambda a: bool(a),  
 'var': lambda obj, key=None: obj if key is None else (obj.get(key) if isinstance(obj, dict) else None)  
}  
  
def eval\_logic(expr, data):  
 if isinstance(expr, (int,float,str,bool)) or expr is None:  
 return expr  
 if isinstance(expr, list):  
 return [eval\_logic(x, data) for x in expr]  
 if isinstance(expr, dict):  
 if len(expr)!=1:  
 raise ValueError('Invalid logic obj')  
 (op, args), = expr.items()  
 fn = ALLOWED\_FUNCS.get(op)  
 if fn is None: raise ValueError(f'Op {op} not allowed')  
 if not isinstance(args, list): args=[args]  
 ev = [eval\_logic(a, data) for a in args]  
 return fn(\*ev)  
 raise TypeError('Unsupported type')  
  
def test\_expr(expr, data) -> bool:  
 try:  
 if expr.strip().startswith('{'):  
 obj = json.loads(expr)  
 return bool(eval\_logic(obj, data))  
 # simple pythonic infix subset: a == b etc.  
 if '==' in expr:  
 k,v = [x.strip() for x in expr.split('==',1)]  
 return str(data.get(k)) == v.strip("'\"") if isinstance(data, dict) else False  
 return False  
 except Exception:  
 return False

### 9.6.4 JSONPath Subset (jsonpath.py)

from typing import Any, List, Tuple  
  
# Minimal subset: $, dot, wildcard, recursive descent .., array wildcard [\*]  
  
def select(obj: Any, path: str) -> List[Tuple[str, Any]]:  
 results = []  
 def rec(o, tokens, ptr):  
 if not tokens:  
 results.append((ptr or '/', o)); return  
 tok = tokens[0]  
 rest = tokens[1:]  
 if tok == '$':  
 rec(o, rest, '')  
 elif tok == '\*':  
 if isinstance(o, dict):  
 for k,v in o.items(): rec(v, rest, f"{ptr}/{k}")  
 elif isinstance(o, list):  
 for i,v in enumerate(o): rec(v, rest, f"{ptr}/{i}")  
 elif tok == '..':  
 rec(o, rest, ptr)  
 if isinstance(o, dict):  
 for k,v in o.items(): rec(v, tokens, f"{ptr}/{k}")  
 elif isinstance(o, list):  
 for i,v in enumerate(o): rec(v, tokens, f"{ptr}/{i}")  
 elif tok.endswith('[\*]'):  
 key = tok[:-3]  
 if isinstance(o, dict) and key in o and isinstance(o[key], list):  
 for i,v in enumerate(o[key]): rec(v, rest, f"{ptr}/{key}/{i}")  
 else:  
 if isinstance(o, dict) and tok in o:  
 rec(o[tok], rest, f"{ptr}/{tok}")  
 # tokenize: $.a.b[\*].c or $..valueQuantity  
 tokens = []  
 i=0; s=path  
 while i < len(s):  
 if s[i] == '$': tokens.append('$'); i+=1  
 elif s.startswith('..', i): tokens.append('..'); i+=2  
 elif s[i] == '.': i+=1  
 else:  
 j=i  
 while j < len(s) and s[j] not in '.': j+=1  
 tokens.append(s[i:j]); i=j  
 rec(obj, tokens, '')  
 return results

### 9.6.5 Reporters

**JSON (reporters/json.py)**

import json  
  
def write(findings, out):  
 data = {"summary": summary(findings), "findings": [f.\_\_dict\_\_ for f in findings]}  
 out.write(json.dumps(data, indent=2))  
  
def summary(findings):  
 sev = [f.severity for f in findings]  
 status = 'PASS' if not sev else ('ERROR' if 'ERROR' in sev or 'BLOCK' in sev else ('WARN' if 'WARN' in sev else 'INFO'))  
 return {"status": status, "counts": {s: sev.count(s) for s in ['INFO','WARN','ERROR','BLOCK']}}

**Text (reporters/text.py)**

def write(findings, out):  
 if not findings:  
 out.write('PASS\n'); return  
 for f in findings:  
 out.write(f"{f.severity} {f.ruleId} @ {f.pointer}: {f.message}\n")

**HTML (reporters/html.py)**

from jinja2 import Template  
TPL = Template("""  
<!doctype html><html><head><meta charset="utf-8"><title>TIDeS Report</title>  
<style>body{font-family:system-ui} .ERROR{color:#b00020} .WARN{color:#b08400} .INFO{color:#555}</style>  
</head><body>  
<h1>TIDeS Validation Report</h1>  
<p>Status: <strong>{{ summary.status }}</strong></p>  
<table border="1" cellspacing="0" cellpadding="6">  
<tr><th>Severity</th><th>Rule</th><th>Pointer</th><th>Message</th><th>Links</th></tr>  
{% for f in findings %}  
<tr class="{{f.severity}}"><td>{{f.severity}}</td><td>{{f.ruleId}}</td><td><code>{{f.pointer}}</code></td><td>{{f.message}}</td><td>{% for l in f.links %}<a href="#">{{l}}</a>{% if not loop.last %}, {% endif %}{% endfor %}</td></tr>  
{% endfor %}  
</table>  
</body></html>  
""")  
  
def write(findings, out):  
 from .json import summary  
 out.write(TPL.render(findings=[f.\_\_dict\_\_ for f in findings], summary=summary(findings)))

**SARIF (reporters/sarif.py)**

import json  
  
def write(findings, out):  
 runs = [{  
 "tool": {"driver": {"name": "tides-validator", "version": "1.0.0"}},  
 "results": [{  
 "ruleId": f.ruleId,  
 "level": f.severity.lower(),  
 "message": {"text": f.message},  
 "locations": [{"physicalLocation": {"artifactLocation": {"uri": "bundle.json"}, "region": {"snippet": {"text": f.pointer}}}}]  
 } for f in findings]  
 }]  
 out.write(json.dumps({"version": "2.1.0", "runs": runs}, indent=2))

### 9.6.6 CLI (cli.py) with Typer

import json, sys, typer  
from .engine import Engine, Rule, Finding  
from .reporters import json as rjson, text as rtext, html as rhtml, sarif as rsarif  
  
app = typer.Typer(add\_completion=False)  
  
@app.command()  
def validate(bundle: str = typer.Argument(...), profile: str = typer.Option('A'), rules: str = 'validator/rules/expanded.json', format: str = typer.Option('json'), exit\_on: str = typer.Option('ERROR')):  
 inst = json.load(open(bundle))  
 data = json.load(open(rules))  
 ruleset = [Rule(\*\*r) for r in data['rules']]  
 eng = Engine(ruleset, profile)  
 findings = eng.evaluate(inst)  
 if format=='json': rjson.write(findings, sys.stdout)  
 elif format=='text': rtext.write(findings, sys.stdout)  
 elif format=='html': rhtml.write(findings, sys.stdout)  
 elif format=='sarif': rsarif.write(findings, sys.stdout)  
 status = 'PASS' if not findings else ('ERROR' if any(f.severity in ['ERROR','BLOCK'] for f in findings) else 'WARN')  
 code = 0 if status=='PASS' else (1 if status=='ERROR' else 2)  
 raise typer.Exit(code)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app()

## 9.7 Reference Implementation — Node.js

### 9.7.1 Engine Core (ESM)

import jp from 'jsonpath-plus';  
  
export function evaluate(bundle, rules, profile='A') {  
 const out = [];  
 for (const r of rules.filter(x => !x.profiles || x.profiles.includes(profile))) {  
 const nodes = jp.JSONPath({path: r.when, json: bundle, resultType: 'pointer'});  
 for (const ptr of nodes) {  
 const node = jp.JSONPath({path: ptr.replace('#',''), json: bundle, resultType: 'value'})[0];  
 let ok = true;  
 if (r.test) ok = safeTest(r.test, node);  
 if (r.negate) ok = !ok;  
 if (!ok) out.push({ruleId: r.id, severity: r.severity, pointer: ptr, message: r.desc, links: r.links||[]});  
 }  
 }  
 return out.sort((a,b)=> a.ruleId.localeCompare(b.ruleId));  
}  
function safeTest(expr, node){  
 try { const obj = JSON.parse(expr); return obj['=='] ? obj['=='][0] === obj['=='][1] : true; } catch { return false; }  
}

### 9.7.2 CLI (commander)

#!/usr/bin/env node  
import { program } from 'commander';  
import fs from 'fs';  
import { evaluate } from './engine.js';  
  
program.option('-b, --bundle <path>').option('-r, --rules <path>').option('-p, --profile <A|B|C>', 'A').option('-f, --format <fmt>', 'json');  
program.parse(process.argv);  
const o = program.opts();  
const bundle = JSON.parse(fs.readFileSync(o.bundle));  
const rules = JSON.parse(fs.readFileSync(o.rules)).rules;  
const findings = evaluate(bundle, rules, o.profile);  
if (o.format==='json') console.log(JSON.stringify({findings}, null, 2));  
process.exit(findings.some(f=>['ERROR','BLOCK'].includes(f.severity)) ? 1 : (findings.length?2:0));

## 9.8 HTTP API (OpenAPI 3.1)

openapi: 3.1.0  
info: { title: TIDeS Validator API, version: 1.0.0 }  
paths:  
 /validate:  
 post:  
 summary: Validate a TIDeS bundle  
 requestBody:  
 required: true  
 content:  
 application/json:  
 schema: { $ref: '#/components/schemas/TidesBundle' }  
 responses:  
 '200': { description: Validation report, content: { application/json: { schema: { $ref: '#/components/schemas/Report' } } } }  
components:  
 schemas:  
 TidesBundle: { type: object }  
 Report:  
 type: object  
 properties:  
 summary: { type: object }  
 findings: { type: array, items: { type: object } }

**Normative:** HTTP mode **MUST** respect the same rule pack and determinism; no per‑request mutation of packs.

## 9.9 Exit Codes, Status, Badges

* **Exit codes:** 0=PASS, 1=ERROR/BLOCK, 2=WARN, 3=USAGE.
* **Badges:** A/B/C emitted when all MUST rules pass for selected profile; JSON carries badge: 'A-PASS'.
* **HTML badge** snippet is embedded in HTML reports and publishable to CI artifacts.

## 9.10 Reporting (Formats)

* **JSON**: machine‑readable, includes pointers & links.
* **Text**: CI logs.
* **HTML**: human‑friendly, color‑coded, anchored rules.
* **SARIF**: integrates with code scanning / GitHub Security tab.
* **JUnit (optional)**: single testcase per rule with failures; **MAY** be enabled with --format junit.

## 9.11 DICOM/FHIR/OMOP Plug‑ins

* **DICOM**: parses headers to assert tags; never exports pixel data in reports; **MUST** redact PHI fields in logs.
* **FHIR**: optional invocation of HL7 validator for profile checking; timeouts and offline mode enforced.
* **OMOP**: simple DDL version introspection; can run SELECT checks when DB DSN provided.

**Sandboxing:** Plug‑ins run in the same process but are **pure**; no network or file writes beyond configured roots.

## 9.12 Performance & Scalability

* **Complexity:** O(Nrules × Nselected nodes). Rule selectors pre‑compiled.
* **Concurrency:** CLI --jobs allows parallel rule groups; ordering preserved by stable sort.
* **Memory:** Stream large JSON with iterators; HTML report generated with chunked templates.

**Benchmarks:** Provided in validator/bench/ with synthetic bundles at 1k/10k resources.

## 9.13 Security & Privacy

* **PHI:** Reports never include PHI; pointers show JSONPath, not values (except numeric metrics).
* **No exfiltration:** Network disabled by default; --allow-net required to enable FHIR validation endpoints.
* **Reproducibility:** All dependencies pinned; hashes recorded in provenance of the report.

## 9.14 Configuration & Environment

* tides.yaml (optional):

profile: A  
rules: validator/rules/expanded.json  
policyPacks:  
 - validator/policy/policy-oar-1.0.0.yaml  
report:  
 format: [json, html]  
 outdir: .tides/reports

* **Env vars:** TIDES\_RULES, TIDES\_PROFILE, TIDES\_POLICY\_DIR, NO\_COLOR.

## 9.15 CI Integration (GitHub/GitLab/Azure)

**GitHub Actions**

name: TIDeS Validate  
on: [push, pull\_request]  
jobs:  
 validate:  
 runs-on: ubuntu-latest  
 steps:  
 - uses: actions/checkout@v4  
 - uses: actions/setup-python@v5  
 with: { python-version: '3.11' }  
 - run: pip install -r validator/requirements.txt  
 - run: python -m validator.cli validate examples/case\_pass\_minimal/bundle.json --profile A --format html > report.html  
 - uses: actions/upload-artifact@v4  
 with: { name: tides-report, path: report.html }

**SARIF upload**

- run: python -m validator.cli validate examples/case\_fail\_units/bundle.json --format sarif > report.sarif  
 - uses: github/codeql-action/upload-sarif@v3  
 with: { sarif\_file: report.sarif }

## 9.16 Fixtures & Unit Tests

* **Fixtures** (§11): ten canonical cases produce expected outcomes; CI compares reports byte‑for‑byte.
* **Unit tests** (pytest):

def test\_spec\_version(rule\_engine, bundle\_ok):  
 findings = rule\_engine.evaluate(bundle\_ok)  
 assert not any(f.ruleId=='spec-version' and f.severity=='ERROR' for f in findings)

* **Golden tests**: serialize JSON reports for each fixture and diff against committed goldens.

## 9.17 Extensibility & Custom Rules

* **Custom metrics:** metric: custom with customExpr evaluated by JSONLogic sandbox.
* **Site packs:** include site-rules.json merged with core; **MUST NOT** downgrade core severities.
* **Plugins:** register via entrypoints tides\_validator.plugins in Python setup.cfg.

## 9.18 Determinism, Tolerances & Floating Point

* **Comparisons:** Numeric comparisons use strict inequality on canonical units; optional epsilon (--eps 1e-12) **MAY** be used in research.
* **Ordering:** Findings sorted by (severity, ruleId, pointer); HTML preserves this order.

## 9.19 Output Contract (JSON)

{  
 "summary": {"status": "WARN", "counts": {"INFO": 1, "WARN": 2, "ERROR": 0, "BLOCK": 0}, "badge": "B-PASS"},  
 "findings": [  
 {"ruleId":"timing-injection","severity":"ERROR","pointer":"/resources/1/timings","message":"Must include injectionStart","links":["§4.0"]}  
 ],  
 "provenance": {"software":"tides-validator","version":"1.0.0","hash":"sha256:...","rules":"expanded.json#1.0.0"}  
}

**Normative:** summary.status reflects worst severity; badge present only when profile MUST requirements are satisfied.

## 9.20 Developer Tooling

* **Dockerfile** (runtime parity):

FROM python:3.11-slim  
WORKDIR /app  
COPY validator /app/validator  
RUN pip install -r /app/validator/requirements.txt  
ENTRYPOINT ["python","-m","validator.cli","validate"]

* **Pre‑commit hooks**: JSON schema lint, rule pack lint, black/isort.

## 9.21 FAQs

**Q: Can the validator auto‑fix issues?**  
A: No. It is read‑only by design. Emit actionable messages and pointers; remediation happens upstream.

**Q: How do we add a new OAR rule without forking?**  
A: Publish a site pack with customExpr; register it via CLI --site-rules. Core rules remain authoritative.

**Q: Can we validate partial bundles?**  
A: Yes with --profile C or with --allow-partial; schema validation will still run on present resources.

## 9.22 Chapter Summary

* A deterministic, offline, layered validator enforces TIDeS **syntax + semantics + interop + policy**.
* Executable rule packs, profiles, safe expression evaluation, rich reporters, and CI harness yield repeatable conformance and badges.

**End of Chapter 9 (Normative).**

# TIDeS Handbook — Chapter 10

## Conformance & Profiles (Must‑Support, Badging, Certification)

**Purpose.** Define the **authoritative conformance program** for TIDeS: profile definitions (A/B/C), Must‑Support matrices, machine‑readable capability statements, badging and certification, test and coverage criteria, waiver governance, reporting, and reference code to compute **PASS/WARN/ERROR** outcomes and issue cryptographically signed badges.

**Audience.** Implementers, QA leads, vendors, site admins, auditors, regulators.

**Outcome.** You will (a) declare your product/site capabilities, (b) select and satisfy a target profile, (c) run the conformance validator and receive a badge, (d) publish capability artifacts (FHIR/DICOM/OMOP), and (e) maintain compliance through change control.

**Normative keywords:** **MUST**, **SHOULD**, **MAY** (RFC 2119/8174).

## 10.0 Scope & Principles (Normative)

1. **Profiles are executable contracts.** Each profile is a reproducible set of MUST/WARN rules across syntax, semantics, interop, and policy (Chapters 1–9).
2. **Badges encode evidence.** Badges are issued **only** by the reference validator when all MUSTs pass for the selected profile; they are **signed** and **verifiable offline**.
3. **Must‑Support ≠ Required Field.** If a Must‑Support element is absent, the producer **MUST** demonstrate that it was not applicable or record a structured notPerformedReason.
4. **Traceability.** Every profile requirement maps 1‑to‑1 to rule IDs and spec clauses.

## 10.1 Profiles Overview (Authoritative)

**Profile A — Clinical‑Full.** - **Intent:** Clinical theranostics with voxel dose, safety policy enforcement, full interop (DICOM/SEG/RTDOSE + FHIR DiagnosticReport).  
- **Audience:** Hospitals, EHR integration, regulators.

**Profile B — Research‑Voxel.** - **Intent:** Research pipelines with voxel dosimetry and reproducible provenance; FHIR recommended, policy optional.  
- **Audience:** Academic labs, trials, algorithm vendors.

**Profile C — Legacy‑Organ.** - **Intent:** Organ‑level dosimetry without voxel artifacts (seg/dose maps optional); timing/units/provenance still enforced.  
- **Audience:** Legacy systems, retrospective analyses.

## 10.2 Must‑Support Matrix (Normative)

| Capability | A (Clinical‑Full) | B (Research‑Voxel) | C (Legacy‑Organ) | Rules |
| --- | --- | --- | --- | --- |
| specVersion present | **MUST** | **MUST** | **MUST** | spec-version |
| injectionStart (ISO‑8601) | **MUST** | **MUST** | **MUST** | timing-injection |
| Frame list & fitWindow\_s | **SHOULD** | **SHOULD** | **MAY** | sampling-adequacy |
| UCUM units (Gy, Gy/s) | **MUST** | **MUST** | **MUST** | unit-ucum, unit-dose-gy, unit-doserate |
| Voxel dose (RTDOSE) + FoR | **MUST** | **MUST** | **N/A** | registration-fuid, rtdose-for-match |
| SEG coded labels | **MUST** | **SHOULD** | **MAY** | seg-coded-meanings |
| Provenance (software + hash) | **MUST** | **MUST** | **SHOULD** | provenance |
| Safety policy evaluation | **MUST** | **SHOULD** | **SHOULD** | policy-declared, constraint-\* |
| FHIR DiagnosticReport | **MUST** | **SHOULD** | **MAY** | fhir-profile-declared |
| OMOP export rows | **SHOULD** | **SHOULD** | **MAY** | omop-ddl-version |

**Normative:** Failure of any **MUST** yields **ERROR/BLOCK** → no badge for that profile.

## 10.3 Profile Definitions (Machine‑Readable)

**Schema:** /schemas/1.0.0/profiles/profile-\*.schema.json (Chapter 8).  
**Overlay:** Each profile is a JSON object listing applicable rule IDs and severity floors.

{  
 "$id": "https://tides.org/profiles/1.0.0/A.json",  
 "name": "A",  
 "title": "Clinical-Full",  
 "rules": {  
 "MUST": ["spec-version","timing-injection","unit-ucum","unit-dose-gy","registration-fuid","provenance","constraint-fail:absent","fhir-profile-declared"],  
 "SHOULD": ["sampling-adequacy","seg-coded-meanings","omop-ddl-version"],  
 "MAY": []  
 }  
}

**Normative:** A profile **MUST NOT** reduce severity of a core rule below its baseline; overlays may raise severities.

## 10.4 Capability Statements (Producer/Consumer)

### 10.4.1 Producer Capability Manifest (capability.json)

{  
 "product": {"name":"AcmeDosimetry","version":"3.2.1"},  
 "tidesSpec": "1.0.0",  
 "profiles": ["A","B"],  
 "supports": {  
 "dicom": ["SEG","RTDOSE","SR-WV","SpatialRegistration"],  
 "fhir": ["DiagnosticReport","Observation(AbsorbedDose)"],  
 "omop": true  
 },  
 "limits": {"maxVoxelGrid": [512,512,512], "kernelSupportVox": 63},  
 "security": {"offline": true, "phiLogging": false}  
}

### 10.4.2 Consumer Requirement Manifest (requirements.json)

{"targetProfile":"A","waivers":[{"ruleId":"seg-coded-meanings","reason":"Legacy SEG without codes; remediation scheduled Q4"}]}

**Validator:** checks compatibility and flags unmet capabilities prior to full validation.

## 10.5 Badge Lifecycle (Signed Badges)

### 10.5.1 Badge Data Model

{  
 "badge": "A-PASS",  
 "specVersion": "1.0.0",  
 "profile": "A",  
 "bundleHash": "sha256-...",  
 "rulesVersion": "1.0.0",  
 "issuedAt": "2025-09-25T12:00:33Z",  
 "issuer": "urn:tides:authority:ref-validator",  
 "findingsDigest": "sha256-...",  
 "signature": {"alg":"Ed25519","keyId":"did:key:z6Mkh...","value":"base64..."}  
}

### 10.5.2 Reference Signing (Python)

from nacl.signing import SigningKey, VerifyKey  
from nacl.encoding import Base64Encoder  
import json, hashlib, base64  
  
def badge\_sign(report\_json: dict, sk\_b64: str) -> dict:  
 sk = SigningKey(Base64Encoder.decode(sk\_b64))  
 payload = {  
 "badge": report\_json.get("summary",{}).get("badge"),  
 "specVersion": "1.0.0",  
 "profile": report\_json.get("profile","A"),  
 "bundleHash": report\_json.get("provenance",{}).get("bundleHash",""),  
 "rulesVersion": report\_json.get("provenance",{}).get("rules","1.0.0"),  
 "issuedAt": report\_json.get("provenance",{}).get("issuedAt",""),  
 "issuer": "urn:tides:authority:ref-validator",  
 "findingsDigest": hashlib.sha256(json.dumps(report\_json.get("findings",[]), sort\_keys=True).encode()).hexdigest()  
 }  
 msg = json.dumps(payload, separators=(',',':')).encode()  
 sig = sk.sign(msg).signature  
 payload["signature"] = {"alg":"Ed25519","keyId":"did:key:PLACEHOLDER","value": base64.b64encode(sig).decode()}  
 return payload  
  
def badge\_verify(badge: dict, vk\_b64: str) -> bool:  
 vk = VerifyKey(Base64Encoder.decode(vk\_b64))  
 sig = base64.b64decode(badge["signature"]["value"])  
 msg = json.dumps({k:badge[k] for k in badge if k!="signature"}, separators=(',',':')).encode()  
 try:  
 vk.verify(msg, sig); return True  
 except Exception:  
 return False

### 10.5.3 CLI Snippets

$ tides validate bundle.json --profile A --format json > report.json  
$ tides badge sign --in report.json --out badge.json --key ~/.tides/ed25519.sk  
$ tides badge verify --in badge.json --pub ~/.tides/ed25519.vk

## 10.6 Waivers & Exceptions (Governance)

* **Waiver request** includes { ruleId, scope (study/site/product), reason, compensatingControls, expiry }.
* **Scopes:**
  + *Study‑specific* (single bundle): Validator marks as WAIVED; badge **MAY** still be issued if all non‑waived MUSTs pass.
  + *Site‑level* (temporary): Requires signed site policy; expires within ≤90 days.
  + *Product‑level* (rare): Requires balloted change or strong justification.
* **Transparency:** Waivers are embedded in report.json and in the signed badge payload (waiverDigest).

**Validator behavior:** - ERROR → WAIVED when a valid waiver matches; severity is lowered only for badge determination, not erased from findings.

## 10.7 Coverage, Evidence, and Traceability

* **Rule coverage:** % of applicable rules exercised by fixtures/bundle.
* **Data coverage:** % of Must‑Support elements present and populated.
* **Trace matrix:** CSV linking ruleId ↔ spec section ↔ schema path ↔ fixture.
* **Evidence archive:** Store bundle.json, report.json, badge.json, and HTML report with SHA‑256 hashes.

**Reference CSV (excerpt):**

ruleId,section,schemaPointer,fixtures  
spec-version,§0,/specVersion,case\_pass\_minimal  
registration-fuid,§3.1,/resources[\*].frameOfReferenceUID,case\_registration\_ok  
constraint-fail,§6.1-6.3,/policyResults/findings[\*],case\_safety\_violation\_kidney

## 10.8 Profile Selection & Negotiation

**Algorithm (site onboarding):** 1. Read producer capability.json and consumer requirements.json. 2. Compute feasible set {A,B,C}; prefer highest profile that meets requirements. 3. Emit negotiation artifact with selected profile and gaps.

**Pseudocode (TypeScript):**

export function selectProfile(cap: Capability, req: Requirements): {selected: 'A'|'B'|'C', gaps: string[]} {  
 const order: ('A'|'B'|'C')[] = ['A','B','C'];  
 for (const p of order) {  
 if (!cap.profiles.includes(p)) continue;  
 const gaps: string[] = [];  
 if (p==='A' && !cap.supports.fhir?.includes('DiagnosticReport')) gaps.push('FHIR DiagnosticReport');  
 if (p!=='C' && !cap.supports.dicom?.includes('RTDOSE')) gaps.push('DICOM RTDOSE');  
 if (gaps.length===0) return {selected: p, gaps};  
 }  
 return {selected: 'C', gaps: ['Fallback to C']};  
}

## 10.9 Conformance Statements (Publishable Artifacts)

### 10.9.1 DICOM Conformance (Extract)

* IODs produced/consumed, transfer syntaxes, private tags (none for TIDeS), spatial registration handling, RTDOSE constraints (units=GY).
* Include a **tag compliance table** keyed to §7.1.2.

### 10.9.2 FHIR CapabilityStatement (R4)

{  
 "resourceType": "CapabilityStatement",  
 "status": "active",  
 "date": "2025-09-25",  
 "format": ["json"],  
 "rest": [{  
 "mode": "server",  
 "resource": [  
 {"type":"DiagnosticReport","profile":["http://tides.org/fhir/StructureDefinition/tides-diagnosticreport-theranostics"]},  
 {"type":"Observation","profile":["http://tides.org/fhir/StructureDefinition/tides-absorbed-dose"]}  
 ]  
 }]  
}

### 10.9.3 OMOP Conformance

* DDL version support (tides\_meta.ddl\_version = 1.0.0).
* ETL completeness metrics (rows per study, ROI coverage).

## 10.10 Connectathon & IHE‑Style Testing

* **Scenarios:**
  1. Export DICOM (PET/CT+SEG) → Compute PK/Dose → RTDOSE → FHIR → Validate A.
  2. Import foreign kernel/meta → Reproduce dose within tolerance (1e‑4) → Validate B.
  3. Legacy CSV organ summaries → Build DoseReport → Validate C.
* **Monitors:** live validator with projector, QR codes to HTML reports.

## 10.11 CI Gates & Release Criteria

* **Pre‑merge:** All fixtures must pass their expected outcomes; golden reports diff clean.
* **Release:** Rule pack version bump, changelog, signed tag, DOI mint; badges regenerated for fixtures.

**GitHub rule (YAML):**

- run: python -m validator.cli validate examples/case\_pass\_minimal/bundle.json --profile A --format json > .out/minimal.json  
- run: jq -e '.summary.status=="PASS"' .out/minimal.json

## 10.12 Accessibility & Internationalization

* **Reports:** color‑contrast compliant; language packs for HTML reporter (en, fr, de, es).
* **Units:** always UCUM; local display strings may be translated.

## 10.13 Deprecation & Compatibility

* **Minor versions** may add OPTIONAL elements; **MUST NOT** introduce new MUSTs mid‑cycle.
* **Major upgrades** (2.0.0) ship parallel schemas/profiles; migration guide and dual‑validation window (≥6 months).
* **Deprecated rules** remain for ≥1 minor with severity: INFO before removal.

## 10.14 Reference Code — Conformance Orchestrator

### 10.14.1 Python Orchestrator

import json, hashlib, time, pathlib  
from validator.cli import validate as run\_validate  
from validator.engine import Engine, Rule  
  
def conformance\_run(bundle\_path: str, profile: str, rules\_path: str) -> dict:  
 with open(bundle\_path) as f: bundle = json.load(f)  
 with open(rules\_path) as f: rules = [Rule(\*\*r) for r in json.load(f)['rules']]  
 eng = Engine(rules, profile)  
 findings = eng.evaluate(bundle)  
 bundle\_hash = 'sha256-' + hashlib.sha256(json.dumps(bundle, sort\_keys=True).encode()).hexdigest()  
 summary = {"status": 'PASS' if not findings else ('ERROR' if any(x.severity in ['ERROR','BLOCK'] for x in findings) else 'WARN'),  
 "badge": f"{profile}-PASS" if not findings or all(x.severity=='WARN' for x in findings) else None}  
 report = {"profile": profile, "summary": summary, "findings": [f.\_\_dict\_\_ for f in findings],  
 "provenance": {"bundleHash": bundle\_hash, "issuedAt": time.strftime('%Y-%m-%dT%H:%M:%SZ')}}  
 return report

### 10.14.2 Node Badge Tool

import fs from 'fs';  
import { evaluate } from './validator/engine.js';  
export function makeBadge(bundlePath, rulesPath, profile) {  
 const bundle = JSON.parse(fs.readFileSync(bundlePath));  
 const rules = JSON.parse(fs.readFileSync(rulesPath)).rules;  
 const findings = evaluate(bundle, rules, profile);  
 const passMust = findings.filter(f=>f.severity==='ERROR' || f.severity==='BLOCK').length===0;  
 const badge = passMust ? `${profile}-PASS` : null;  
 return { profile, findings, summary: { badge, status: badge? 'PASS':'ERROR' } };  
}

## 10.15 TIDeS‑CHK‑10 — Conformance Readiness

* Target profile chosen; capability/requirement manifests exchanged.
* All MUST rules pass; WARNs documented.
* Badge issued and signed; evidence archived with hashes.
* CI gates enforcing fixtures.
* Waiver policy documented (if any), with expiry dates.
* Conformance statements published (DICOM, FHIR, OMOP).

## 10.16 FAQs

**Q: Can we publish a badge without sharing the bundle?**  
A: Yes—badge contains a digest of the findings and the bundle hash; auditors can reproduce with the same inputs.

**Q: Are WARNs acceptable for A‑PASS?**  
A: Yes; WARNs do not block badges. Any ERROR/BLOCK does.

**Q: How do we handle multi‑site deployments?**  
A: Issue a site‑level badge per installation, each with its own policy packs and capability statements.

## 10.17 Chapter Summary

* Profiles A/B/C are **executable** and enforced via the validator.
* Badges are **signed evidence** of conformance.
* Capability/requirement manifests, waivers, coverage, and CI gates keep implementations honest and reproducible.

**End of Chapter 10 (Normative).**

# TIDeS Handbook — Chapter 11

## Reference Data & Fixtures (Canonical Set + Generators)

**Purpose.** Ship a complete, **runnable** catalog of canonical fixtures with gold‑standard reports so every site and vendor can reproduce TIDeS results byte‑for‑byte. This chapter defines the directory structure, data contracts, synthetic data generators, expected validator outcomes, and CI wiring. It covers the 10 baseline cases listed in the blueprint and extends them with parameterized generators to produce stress and corner‑case corpora.

**Audience.** QA engineers, pipeline developers, vendors, auditors, connectathon monitors.

**Outcome.** You will (a) generate fixtures deterministically, (b) validate them with the reference validator, (c) compare against goldens (JSON/TXT/HTML/SARIF), (d) plug them into your CI/CD, and (e) extend with site‑specific variants without breaking canonical expectations.

**Normative keywords:** **MUST**, **SHOULD**, **MAY**.

## 11.0 Repository Layout (Authoritative)

TIDES/  
 examples/  
 case\_pass\_minimal/  
 case\_fail\_units/  
 case\_registration\_ok/  
 case\_registration\_link\_missing/  
 case\_sampling\_inadequate/  
 case\_safety\_violation\_kidney/  
 case\_provenance\_missing/  
 case\_ucum\_wrong\_unit\_text/  
 case\_flash\_flag\_coverage/  
 case\_legacy\_profile\_C/  
 tools/  
 gen/  
 mkbundle.py # main generator CLI (deterministic)  
 synth\_data.py # voxel/ROI/dose synth  
 fhir\_snips.py # FHIR emit helpers  
 dicom\_stub.py # DICOM header stubs  
 policy.py # loads policy packs  
 validate\_all.py # runs validator on all fixtures  
 validator/  
 rules/expanded.json # copied from Ch.9  
 schemas/ # from Ch.8  
 .golden/ # canonical outputs per case  
 case\_\*/  
 report.json  
 report.txt  
 report.html  
 report.sarif

**Normative:** Canonical fixtures and goldens **MUST** be committed and versioned with the spec. Generators **MUST** be deterministic given the same inputs.

## 11.1 Canonical Case Index (10 Baseline Fixtures)

Each case directory MUST contain: bundle.json, policy.yaml (when relevant), report.json, report.txt, report.html, report.sarif, README.md.

| # | Case ID | Intent | Expected Status | Key Rules Exercised |
| --- | --- | --- | --- | --- |
| 1 | case\_pass\_minimal | Minimal A‑conformant bundle with correct units/timing/FoR | **A‑PASS** | spec-version, timing-injection, unit-ucum, registration-fuid, provenance |
| 2 | case\_fail\_units | Wrong dose units (mGy) | **FAIL** (ERROR) | unit-dose-gy |
| 3 | case\_registration\_ok | Proper FoR and registrations | **PASS** | registration-fuid, rtdose-for-match |
| 4 | case\_registration\_link\_missing | Missing FoR link | **FAIL** | registration-fuid |
| 5 | case\_sampling\_inadequate | Too few frames for model | **PASS with WARN** | sampling-adequacy |
| 6 | case\_safety\_violation\_kidney | Kidney Dmean exceeds 23 Gy | **FAIL** | constraint-fail |
| 7 | case\_provenance\_missing | No software/hash | **FAIL** | provenance |
| 8 | case\_ucum\_wrong\_unit\_text | Non‑UCUM unit symbol | **FAIL** | unit-ucum |
| 9 | case\_flash\_flag\_coverage | Correct dose‑rate & flashCoverage\_pc populated | **PASS** | unit-doserate, flash reporting |
| 10 | case\_legacy\_profile\_C | Organ‑only legacy dataset | **C‑PASS** | Profile C MUSTs only |

## 11.2 Fixture Contracts (What goes in each folder)

Each examples/case\_\* folder **MUST** include:

* bundle.json — A TIDeS Bundle conforming to Chapter 8.
* policy.yaml — If the case depends on safety rules (e.g., #6), include the exact policy pack.
* report.\* — Outputs from the reference validator (JSON/TXT/HTML/SARIF).
* README.md — Human explanation with rule trace.
* notes.json — Optional metadata (seed, generator parameters, hashes).

## 11.3 Deterministic Synthetic Data (Executables)

### 11.3.1 tools/gen/synth\_data.py

import numpy as np  
from dataclasses import dataclass  
  
@dataclass(frozen=True)  
class RNG:  
 seed: int  
 def np(self):  
 rng = np.random.default\_rng(self.seed)  
 return rng  
  
def make\_grid(shape=(64,64,32), spacing=(3.0,3.0,3.0)):  
 return {  
 'size': list(map(int, shape)),  
 'spacing\_mm': list(map(float, spacing)),  
 'offsets\_mm': [0.0,0.0,0.0]  
 }  
  
def roi\_mask(shape, center, radius):  
 z,y,x = np.indices(shape)  
 cz,cy,cx = center  
 r = radius  
 dist2 = (z-cz)\*\*2 + (y-cy)\*\*2 + (x-cx)\*\*2  
 return (dist2 <= r\*\*2)  
  
def synth\_dose(shape, baseGy=10.0, gradient=0.002):  
 z,y,x = np.indices(shape)  
 D = baseGy + gradient\*(x+y+z)  
 return D.astype(np.float32)  
  
def apply\_roi\_stats(D, mask):  
 vals = D[mask]  
 return float(vals.mean()), float(vals.max())

### 11.3.2 tools/gen/fhir\_snips.py

def obs\_absorbed\_dose(code, valueGy):  
 return {  
 'resourceType':'Observation',  
 'meta': {'profile':['http://tides.org/fhir/StructureDefinition/tides-absorbed-dose']},  
 'status':'final',  
 'code': {'coding':[{'system':'http://loinc.org','code':'89469-4','display':'Absorbed dose'}]},  
 'valueQuantity': {'value': valueGy, 'system':'http://unitsofmeasure.org', 'code': 'Gy'},  
 'bodySite': {'coding':[{'system':'http://snomed.info/sct','code': code}]}  
 }

### 11.3.3 tools/gen/dicom\_stub.py

def rt\_dose\_stub(fuid='1.2.826.0.1.3680043.2.1125.1', gy=True):  
 return {  
 'DoseUnits': 'GY' if gy else 'RELATIVE',  
 'ReferencedFrameOfReferenceSequence':[{'FrameOfReferenceUID': fuid}],  
 'DoseGridScaling': 1.0,  
 'GridFrameOffsetVector': [float(i) for i in range(0,10)]  
 }

### 11.3.4 tools/gen/policy.py

import yaml  
  
def default\_policy():  
 return yaml.safe\_load('''  
name: policy-oar  
version: 1.0.0  
specMajor: 1  
notes: Default adult limits  
oars:  
 - { roiCode: "64033007", roiCodeSystem: SNOMED, roiCodeText: kidney, metric: Dmean\_Gy, limit: 23, warnThreshold: 20, unit: Gy, aggregation: cumulative }  
 - { roiCode: "74281007", roiCodeSystem: SNOMED, roiCodeText: liver, metric: Dmean\_Gy, limit: 30, warnThreshold: 27, unit: Gy, aggregation: cumulative }  
 - { roiCode: "marrow", roiCodeSystem: LOCAL, roiCodeText: red marrow, metric: Dmean\_Gy, limit: 2, warnThreshold: 1.8, unit: Gy, aggregation: cumulative }  
''')

## 11.4 Bundle Generator (CLI)

### 11.4.1 tools/gen/mkbundle.py

#!/usr/bin/env python3  
import json, argparse, uuid, hashlib, os, math  
from pathlib import Path  
from synth\_data import RNG, make\_grid, synth\_dose, roi\_mask, apply\_roi\_stats  
from policy import default\_policy  
  
FUID = '1.2.826.0.1.3680043.2.1125.1'  
  
def urn(kind, flavor=None):  
 u = str(uuid.uuid4())  
 return f"urn:tides:{kind}:{flavor+':'+u if flavor else u}"  
  
def base\_provenance():  
 return {"software":"tides-cli","version":"1.0.0","hash":"deadbeef","inputs":["dicom:stub"],"policyPacks":[{"name":"policy-oar","version":"1.0.0","specMajor":1}]}  
  
def mk\_case(case\_id: str, params: dict) -> dict:  
 shape = tuple(params.get('shape',(32,32,16)))  
 grid = make\_grid(shape)  
 D = synth\_dose(shape, baseGy=params.get('baseGy',10.0), gradient=params.get('gradient',0.0))  
 # ROIs  
 kidney\_mask = roi\_mask(shape, (shape[0]//2, shape[1]//3, shape[2]//2), radius=max(shape)//6)  
 liver\_mask = roi\_mask(shape, (shape[0]//2, shape[1]//2, shape[2]//2), radius=max(shape)//5)  
 kmean,kmax = apply\_roi\_stats(D, kidney\_mask)  
 lmean,lmax = apply\_roi\_stats(D, liver\_mask)  
  
 dosemap\_id = urn('dosemap','177Lu')  
 subject\_id = urn('subject')  
 bundle = {  
 "specVersion":"1.0.0",  
 "id": urn('bundle','canon'),  
 "study": urn('study'),  
 "resources": [  
 {"resourceType":"TidesImaging","id": urn('imaging'),"frameOfReferenceUID":FUID,  
 "timings":{"injectionStart":"2025-01-01T10:00:00Z","frames":[{"tMid\_s":600,"duration\_s":120}]},  
 "dicomSeries":["dicomweb:..."]},  
 {"resourceType":"TidesDoseMap","id": dosemap\_id, "frameOfReferenceUID":FUID,  
 "kernel":{"formalism":"S-value","nuclide":"177Lu","medium":"soft","grid\_mm":[3,3,3],"version":"v1"},  
 "source": urn('pkmodel','monoexp'), "flashCoverage\_pc": params.get('flash\_pc',0.0)},  
 {"resourceType":"TidesDoseReport","id": urn('dosereport'), "subject": subject\_id,  
 "series":[{"doseMap": dosemap\_id, "organSummaries": [  
 {"roiCode":"64033007","roiCodeText":"kidney","Dmean\_Gy": round(kmean,1), "Dmax\_Gy": round(kmax,1), "uncertainty\_pc": 5.0, "uncertaintyMethod":"delta-method"},  
 {"roiCode":"74281007","roiCodeText":"liver","Dmean\_Gy": round(lmean,1), "Dmax\_Gy": round(lmax,1), "uncertainty\_pc": 9.1, "uncertaintyMethod":"delta-method"}  
 ]}]  
 }  
 ],  
 "provenance": base\_provenance()  
 }  
 # Case‑specific edits  
 if case\_id == 'case\_fail\_units':  
 # Wrong unit will be represented in a FHIR Observation during interop tests, but here alter report to simulate later check  
 bundle['resources'][2]['series'][0]['organSummaries'][0]['Dmean\_Gy'] = 18000 # mGy masquerading as Gy  
 if case\_id == 'case\_registration\_link\_missing':  
 bundle['resources'][1]['frameOfReferenceUID'] = ''  
 if case\_id == 'case\_sampling\_inadequate':  
 bundle['resources'][0]['timings']['frames'] = [{"tMid\_s": 600, "duration\_s": 120}] # monoexp < 3 → WARN  
 if case\_id == 'case\_safety\_violation\_kidney':  
 bundle['resources'][2]['series'][0]['organSummaries'][0]['Dmean\_Gy'] = 24.2  
 if case\_id == 'case\_provenance\_missing':  
 bundle['provenance'].pop('software', None)  
 bundle['provenance'].pop('hash', None)  
 if case\_id == 'case\_ucum\_wrong\_unit\_text':  
 # Inject a bad FHIR Observation unit for interop test via notes  
 bundle.setdefault('notes',{})['bad\_fhir\_unit'] = True  
 if case\_id == 'case\_flash\_flag\_coverage':  
 bundle['resources'][1]['flashCoverage\_pc'] = params.get('flash\_pc', 3.1)  
 if case\_id == 'case\_legacy\_profile\_C':  
 # Remove voxel map by setting organ‑only report  
 bundle['resources'] = [bundle['resources'][2]]  
 return bundle  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 ap = argparse.ArgumentParser()  
 ap.add\_argument('--case', required=True)  
 ap.add\_argument('--out', required=True)  
 args = ap.parse\_args()  
 params = {}  
 if args.case == 'case\_flash\_flag\_coverage': params['flash\_pc'] = 4.0  
 bundle = mk\_case(args.case, params)  
 Path(args.out).write\_text(json.dumps(bundle, indent=2))  
 print(f"Wrote {args.out}")

## 11.5 Canonical Bundles (Abbreviated Examples)

### 11.5.1 case\_pass\_minimal/bundle.json (extract)

{  
 "specVersion": "1.0.0",  
 "id": "urn:tides:bundle:canon:...",  
 "study": "urn:tides:study:...",  
 "resources": [  
 {"resourceType":"TidesImaging","id":"urn:tides:imaging:...","frameOfReferenceUID":"1.2.826.0.1.3680043.2.1125.1","timings":{"injectionStart":"2025-01-01T10:00:00Z","frames":[{"tMid\_s":600,"duration\_s":120}]}},  
 {"resourceType":"TidesDoseMap","id":"urn:tides:dosemap:177Lu:...","frameOfReferenceUID":"1.2.826.0.1.3680043.2.1125.1","kernel":{"formalism":"S-value","nuclide":"177Lu","medium":"soft","grid\_mm":[3,3,3],"version":"v1"},"source":"urn:tides:pkmodel:monoexp:...","flashCoverage\_pc":0.0},  
 {"resourceType":"TidesDoseReport","id":"urn:tides:dosereport:...","subject":"urn:tides:subject:...","series":[{"doseMap":"urn:tides:dosemap:177Lu:...","organSummaries":[{"roiCode":"74281007","roiCodeText":"liver","Dmean\_Gy":18.2,"Dmax\_Gy":21.5,"uncertainty\_pc":9.1,"uncertaintyMethod":"delta-method"}]}]}  
 ],  
 "provenance": {"software":"tides-cli","version":"1.0.0","hash":"deadbeef","inputs":["dicom:stub"],"policyPacks":[{"name":"policy-oar","version":"1.0.0","specMajor":1}]}  
}

### 11.5.2 case\_fail\_units/bundle.json (difference)

* Same as minimal but **kidney Dmean\_Gy** set to 18000 to trigger unit checks downstream (interpreted as error by policy/unit rules).

### 11.5.3 case\_registration\_link\_missing/bundle.json

* TidesDoseMap.frameOfReferenceUID = "".

### 11.5.4 case\_sampling\_inadequate/bundle.json

* frames = [ {tMid\_s:600, duration\_s:120} ] only.

### 11.5.5 case\_safety\_violation\_kidney/bundle.json

* kidney Dmean\_Gy = 24.2 (> 23 Gy limit) → constraint-fail.

### 11.5.6 case\_provenance\_missing/bundle.json

* provenance.software and provenance.hash absent.

### 11.5.7 case\_ucum\_wrong\_unit\_text

* Add notes.bad\_fhir\_unit=true; generator for FHIR export will create a bad valueQuantity.code ("Gray" instead of "Gy").

### 11.5.8 case\_flash\_flag\_coverage

* TidesDoseMap.flashCoverage\_pc = 4.0.

### 11.5.9 case\_legacy\_profile\_C

* Only TidesDoseReport present; no voxel/SEG.

## 11.6 Golden Reports (Canonical Outputs)

For each case, store goldens under .golden/case\_\*/report.\*. The validator **MUST** reproduce these byte‑for‑byte except for stable metadata fields (issuedAt, hashes) which are normalized before diff.

### 11.6.1 Example report.json (PASS + WARN)

{  
 "summary": {"status": "WARN", "counts": {"INFO": 0, "WARN": 1, "ERROR": 0, "BLOCK": 0}, "badge": "A-PASS"},  
 "findings": [  
 {"ruleId":"sampling-adequacy","severity":"WARN","pointer":"/resources/0/timings","message":"Frames must satisfy model minima","links":["§4.0"]}  
 ]  
}

### 11.6.2 Example report.txt

WARN sampling-adequacy @ /resources/0/timings: Frames must satisfy model minima

### 11.6.3 Example report.html

* Rendered table with 1 WARN (see Chapter 9 HTML template). Store exact file.

### 11.6.4 Example report.sarif

* SARIF payload with a single level:"warning".

## 11.7 Fixture READMEs (Template)

examples/case\_X/README.md **MUST** follow this template:

# Case: <id>  
  
\*\*Intent:\*\* <one sentence>  
  
\*\*Inputs:\*\*  
- Bundle: `bundle.json`  
- Policy: `policy.yaml` (if applicable)  
  
\*\*Expected Validator Outcome:\*\* PASS | WARN | ERROR (Badge: A/B/C-PASS or none)  
  
\*\*Rules Exercised:\*\*  
- rule-id: short explanation → section link  
  
\*\*Reproduction:\*\*  
```bash  
python tools/gen/mkbundle.py --case <id> --out examples/<id>/bundle.json  
python -m validator.cli validate examples/<id>/bundle.json --profile <A|B|C> --format json > examples/<id>/report.json

**Notes:** - Deterministic seed; no PHI.

---  
  
## 11.8 CI Harness (Validate All)  
  
### 11.8.1 `tools/validate\_all.py`  
```python  
import subprocess, json, sys, hashlib, os  
from pathlib import Path  
  
ROOT = Path(\_\_file\_\_).resolve().parents[1]  
CASES = [p.name for p in (ROOT/'examples').iterdir() if p.is\_dir() and p.name.startswith('case\_')]  
  
FAIL = 0  
for c in sorted(CASES):  
 bundle = ROOT/'examples'/c/'bundle.json'  
 out = ROOT/'.out'/f'{c}.json'  
 out.parent.mkdir(exist\_ok=True)  
 profile = 'A' if c != 'case\_legacy\_profile\_C' else 'C'  
 cmd = [sys.executable, '-m', 'validator.cli', 'validate', str(bundle), '--profile', profile, '--format', 'json']  
 res = subprocess.run(cmd, capture\_output=True, text=True)  
 if res.returncode not in (0,2):  
 print(f'[FAIL] {c}: exit {res.returncode}')  
 print(res.stdout); print(res.stderr)  
 FAIL += 1; continue  
 out.write\_text(res.stdout)  
 # Compare to golden (normalize issuedAt if present)  
 golden = ROOT/'.golden'/c/'report.json'  
 got = json.loads(res.stdout)  
 want = json.loads(golden.read\_text())  
 got.get('provenance',{}).pop('issuedAt', None)  
 want.get('provenance',{}).pop('issuedAt', None)  
 if got != want:  
 print(f'[DIFF] {c}: report mismatch')  
 FAIL += 1  
  
sys.exit(FAIL)

### 11.8.2 GitHub Actions Snippet

- name: Generate fixtures  
 run: |  
 python tools/gen/mkbundle.py --case case\_pass\_minimal --out examples/case\_pass\_minimal/bundle.json  
 python tools/gen/mkbundle.py --case case\_fail\_units --out examples/case\_fail\_units/bundle.json  
 # ... repeat for all cases ...  
- name: Validate all  
 run: python tools/validate\_all.py

## 11.9 Extending the Canonical Set (Parametric Generator)

You **MAY** add derived corpora using a **case matrix** CSV:

### 11.9.1 examples/case\_matrix.csv

case\_id,baseGy,gradient,flash\_pc  
case\_pass\_minimal,10.0,0.0,0.0  
case\_sampling\_inadequate,10.0,0.0,0.0  
case\_flash\_flag\_coverage,12.0,0.001,3.1

### 11.9.2 Driver (Python)

import csv, subprocess, sys  
from pathlib import Path  
with open('examples/case\_matrix.csv') as f:  
 for row in csv.DictReader(f):  
 out = Path('examples')/row['case\_id']/'bundle.json'  
 args = ['python','tools/gen/mkbundle.py','--case',row['case\_id'],'--out',str(out)]  
 subprocess.run(args, check=True)

## 11.10 Data Integrity & Hashing

* **Bundle hash:** store sha256 of bundle.json in report.json.provenance.bundleHash.
* **Binary attachments:** if any URIs point to binaries, include their SHA‑256 in attachments[].hash.
* **Golden freeze:** Prior to release, recompute hashes and commit.

**Python helper:**

import hashlib, json  
from pathlib import Path  
p = Path('examples/case\_pass\_minimal/bundle.json')  
print('sha256-'+hashlib.sha256(p.read\_bytes()).hexdigest())

## 11.11 Privacy & Syntheticism

All canonical fixtures are **synthetic** and **PHI‑free**. Patient IDs are URNs, not real identifiers. DICOM stubs omit PHI fields and include only the tags required for interop tests.

## 11.12 Troubleshooting Matrix

| Symptom | Likely Cause | Fix |
| --- | --- | --- |
| registration-fuid ERROR on case 1 | Generator mismatch or manual edit | Regenerate bundle; ensure FoR present |
| unit-ucum ERROR on case 8 not triggered | FHIR export not run | Ensure your interop test emits Observation with bad unit |
| Golden diff | Using modified rule pack | Pin validator/rules/expanded.json to 1.0.0 |

## 11.13 Traceability (Rules ↔ Cases)

spec-version → case\_pass\_minimal  
unit-dose-gy → case\_fail\_units  
registration-fuid → case\_registration\_link\_missing  
sampling-adequacy → case\_sampling\_inadequate  
constraint-fail → case\_safety\_violation\_kidney  
provenance → case\_provenance\_missing  
unit-ucum → case\_ucum\_wrong\_unit\_text  
flash-coverage → case\_flash\_flag\_coverage  
profile-C → case\_legacy\_profile\_C

## 11.14 Chapter Summary

* Ten canonical fixtures cover the core conformance surface (syntax, interop, policy).
* Deterministic generators produce PHI‑free bundles with controllable parameters.
* Goldens lock expected outcomes for CI and certification.
* Extensible case matrix supports site‑specific corpora without altering canon.

**End of Chapter 11 (Normative & Executable).**

# TIDeS Handbook — Chapter 12

## Reporting & Templates (Clinical/Research, HTML/PDF, TIDeS‑CHK)

**Purpose.** Define the **authoritative, executable** reporting layer for TIDeS: clinical 2‑page report, research 4‑page report, TIDeS‑CHK checklist, validator report embedding, accessibility/localization, and programmatic generation (HTML → PDF) with signed attachments. This chapter includes data models, templates, and reference code (Python/Node) to render repeatable artefacts for records, manuscripts, and audits.

**Audience.** Clinical physicists and NM physicians, trial coordinators, QA/IT, vendors, regulators.

**Outcome.** You will (a) render the clinical and research reports from one JSON input (**TidesDoseReport + policyResults + provenance**), (b) attach TIDeS‑CHK, (c) embed validator findings, (d) export FHIR DiagnosticReport + PDFs, and (e) archive signed outputs.

**Normative keywords:** **MUST**, **SHOULD**, **MAY**.

## 12.0 Scope & Principles (Normative)

1. **Single‑source rendering.** Reports render from the DoseReport JSON and related artifacts (§8.7) without manual editing.
2. **Deterministic output.** Given the same inputs, report bytes are identical (except for timestamp fields explicitly marked as non‑deterministic).
3. **Clinical brevity, research depth.** Clinical report = **2 pages max** (A4/Letter) with essential metrics and safety; research report = **≤4 pages** with model diagnostics and methods.
4. **Accessibility.** Contrast‑safe, keyboard navigable, screen‑reader labelled; PDF tagged.
5. **Localization.** UI strings are translatable; units remain UCUM.
6. **Redaction.** PHI is excluded by default; local overlay may inject display identifiers (site policy), never stored in artifacts.

## 12.1 Report Data Model (Normative)

**tides-reportdoc.schema.json** (authoritative subset for renderers)

{  
 "$id":"https://tides.org/schemas/1.0.0/tides-reportdoc.schema.json",  
 "$schema":"https://json-schema.org/draft/2020-12/schema",  
 "title":"TIDeS Renderable Report Document",  
 "type":"object",  
 "required":["doseReport","policyResults","provenance"],  
 "properties":{  
 "doseReport": {"$ref":"https://tides.org/schemas/1.0.0/tides-dosereport.schema.json"},  
 "policyResults": {"type":"object"},  
 "provenance": {"$ref":"https://tides.org/schemas/1.0.0/$defs/common.json#/$defs/Provenance"},  
 "site": {"type":"object","properties":{"displayId":{"type":"string"},"logoURI":{"type":"string"},"jurisdiction":{"type":"string"}}},  
 "locale": {"type":"string","default":"en"},  
 "attachments": {"type":"array","items":{"type":"object","properties":{"rel":{"type":"string"},"uri":{"type":"string"},"hash":{"type":"string"}}}},  
 "figures": {"type":"array","items":{"type":"object","properties":{"id":{"type":"string"},"caption":{"type":"string"},"imgURI":{"type":"string"}}}}  
 }  
}

## 12.2 Clinical Report (2‑Page) — Structure & Content (Normative)

**Page 1 — Summary**  
- Header: Site logo (optional), title *“Theranostics Dose Report (TIDeS)”*, specVersion, profile badge.  
- Patient context: **Subject URN only** (PHI‑free), study label, modality (PET/NM/CT/MR), nuclide/agent.  
- Key organ table: ROI, Dmean (Gy), Dmax (Gy), Uncertainty (%), Safety status (PASS/WARN/ERROR) with color glyphs.  
- Policy box: policy name/version, pediatric modifiers applied (if any).  
- Flash box: flashCoverage\_pc and dose‑rate note (see Ch.4).

**Page 2 — Detail**  
- DVH thumbnail (optional) per ROI.  
- Sampling adequacy panel (frames table).  
- Provenance: software/version/hash, inputs (hash‑redacted), parameters.  
- Validator outcome: worst severity, pointer of major findings, QR/link to full HTML validator report.  
- Sign‑off box: preparer, reviewer (initials only), timestamps, jurisdiction tags (GDPR/HIPAA) from §15.

**Normative:** The clinical PDF **MUST NOT** include full PHI; display identifiers are optional and injected by site overlay.

## 12.3 Research Report (≤4 Pages) — Structure & Content (Normative)

* Page 1: Same header + extended organ table with Vx/Dx columns, uncertainty method, policy status.
* Page 2: PK model fits — parameter tables, R²/AIC, residual plots, tail integration details.
* Page 3: Voxel dose maps (orthogonal slices), DVH curves per ROI, radiomics hooks (if present).
* Page 4: Methods appendix (kernel provenance; calibration; partial‑volume/HU→ρ if used).

**Image policy:** all images derived from synthetic or de‑identified sources; lossless and down‑sampled for PDF.

## 12.4 TIDeS‑CHK (18‑Item Manuscript Checklist)

**Form:** One page, checkboxes with rule IDs and brief justifications.  
**Attachment:** Included as separate PDF page or appendix; mirrors §12.8 template.  
**Normative:** Item texts **MUST** map 1:1 to validator rule IDs for traceability.

## 12.5 HTML Template (Authoritative) — Shared for Clinical/Research

**docs/templates/report.html.j2** (Jinja2)

<!doctype html>  
<html lang="{{ locale }}">  
<head>  
<meta charset="utf-8">  
<title>TIDeS Report</title>  
<meta name="viewport" content="width=device-width, initial-scale=1">  
<style>  
 :root { --ok:#1b873f; --warn:#b08400; --err:#b00020; --ink:#222; --muted:#666; }  
 @page { size: A4; margin: 14mm; }  
 body { font-family: system-ui, -apple-system, Segoe UI, Roboto, Ubuntu; color: var(--ink); }  
 header { display:flex; align-items:center; justify-content:space-between; margin-bottom:8mm; }  
 h1 { font-size: 20pt; margin:0; }  
 .muted { color: var(--muted); }  
 table { width:100%; border-collapse: collapse; margin: 4mm 0; }  
 th, td { border-bottom: 1px solid #ddd; padding: 3mm; text-align:left; font-size: 10pt; }  
 .status-ok { color: var(--ok); font-weight:600; }  
 .status-warn { color: var(--warn); font-weight:600; }  
 .status-err { color: var(--err); font-weight:700; }  
 .grid { display:grid; grid-template-columns: 1fr 1fr; gap: 6mm; }  
 .box { border: 1px solid #e5e5e5; border-radius: 8px; padding: 4mm; }  
 footer { position: running(footer); font-size: 9pt; color: var(--muted); margin-top:6mm; }  
 @page { @bottom-center { content: element(footer) } }  
</style>  
</head>  
<body>  
<header>  
 <div>  
 <h1>Theranostics Dose Report (TIDeS)</h1>  
 <div class="muted">Spec {{ doseReport.specVersion }} · Profile {{ badge or '—' }}</div>  
 </div>  
 {% if site.logoURI %}<img src="{{ site.logoURI }}" alt="Site Logo" style="height:24mm">{% endif %}  
</header>  
<section class="grid">  
 <div class="box">  
 <b>Subject:</b> {{ doseReport.subject }}<br>  
 <b>DoseMap:</b> {{ doseReport.series[0].doseMap }}<br>  
 <b>Policy:</b> {{ policyResults.policy.name }} v{{ policyResults.policy.version }}  
 </div>  
 <div class="box">  
 <b>Validator:</b> {{ validator.summary.status }}  
 {% if validator.summary.status == 'ERROR' %}<span class="status-err">●</span>{% elif validator.summary.status == 'WARN' %}<span class="status-warn">●</span>{% else %}<span class="status-ok">●</span>{% endif %}<br>  
 <b>Findings:</b> {{ validator.summary.counts.ERROR }}E/{{ validator.summary.counts.WARN }}W  
 </div>  
</section>  
<h2>Organ Summary</h2>  
<table aria-label="Organ dose summary">  
 <thead><tr><th>ROI</th><th>Dmean (Gy)</th><th>Dmax (Gy)</th><th>Unc. (%)</th><th>Safety</th></tr></thead>  
 <tbody>  
 {% for s in doseReport.series %}  
 {% for o in s.organSummaries %}  
 {% set f = (policyResults.findings | selectattr('roiCode','equalto',o.roiCode) | list | first) %}  
 {% set status = f.decision if f else 'PASS' %}  
 <tr>  
 <td>{{ o.roiCodeText or o.roiCode }}</td>  
 <td>{{ '%.1f' % o.Dmean\_Gy if o.Dmean\_Gy is defined }}</td>  
 <td>{{ '%.1f' % o.Dmax\_Gy if o.Dmax\_Gy is defined }}</td>  
 <td>{{ '%.1f' % o.uncertainty\_pc if o.uncertainty\_pc is defined }}</td>  
 <td class="status-{{ 'ok' if status=='PASS' else ('warn' if status=='WARN' else 'err') }}">{{ status }}</td>  
 </tr>  
 {% endfor %}  
 {% endfor %}  
 </tbody>  
</table>  
<div class="grid">  
 <div class="box">  
 <b>Flash coverage:</b> {{ doseMap.flashCoverage\_pc or 0 }} %  
 </div>  
 <div class="box">  
 <b>Sampling adequacy:</b> {{ samplingAdequacy or 'OK' }}  
 </div>  
</div>  
<h2>Provenance</h2>  
<div class="box">  
 {{ provenance.software }} {{ provenance.version }} · hash {{ provenance.hash }}  
</div>  
<footer>  
 Generated {{ now }} · Jurisdiction: {{ site.jurisdiction or 'unspecified' }} · © TIDeS 1.0.0  
</footer>  
</body>  
</html>

**Normative:** Template variables **MUST** be drawn from the schema in §12.1; unknown variables are disallowed in the reference renderer (fail fast).

## 12.6 CSS Print & Tagging (Accessibility)

* Use ARIA labels on tables and figures.
* All color indicators **MUST** be redundant with textual labels.
* PDF tags: H1/H2, Table, Figure, P.
* Font sizes: body ≥ 9pt; headings ≥ 12pt; line height ≥ 1.25.

## 12.7 Renderer Reference Code

### 12.7.1 Python (Jinja2 + WeasyPrint)

from jinja2 import Environment, FileSystemLoader, StrictUndefined  
from weasyprint import HTML  
import json, datetime, pathlib  
  
def render\_report(doc\_json: dict, template\_dir: str, html\_out: str, pdf\_out: str):  
 env = Environment(loader=FileSystemLoader(template\_dir), autoescape=True, undefined=StrictUndefined)  
 tpl = env.get\_template('report.html.j2')  
 # Derived fields  
 ctx = dict(doc\_json)  
 ctx['now'] = datetime.datetime.utcnow().strftime('%Y-%m-%d %H:%MZ')  
 # Helper deref  
 doseMap = None  
 for r in doc\_json['doseReport']['series']:  
 doseMap = r.get('doseMap')  
 break  
 ctx['doseMap'] = {'flashCoverage\_pc': doc\_json.get('doseMap',{}).get('flashCoverage\_pc', doc\_json.get('policyResults',{}).get('flashCoverage\_pc'))}  
 html = tpl.render(\*\*ctx)  
 pathlib.Path(html\_out).write\_text(html, encoding='utf-8')  
 HTML(string=html).write\_pdf(pdf\_out)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 import sys  
 doc = json.load(open(sys.argv[1]))  
 render\_report(doc, 'docs/templates', 'out/report.html', 'out/report.pdf')

### 12.7.2 Node (Nunjucks + Puppeteer)

import nunjucks from 'nunjucks';  
import fs from 'fs';  
import puppeteer from 'puppeteer';  
  
export async function render(doc, tplDir, htmlOut, pdfOut){  
 const env = nunjucks.configure(tplDir, { autoescape: true });  
 const html = env.render('report.html.njk', doc);  
 fs.writeFileSync(htmlOut, html);  
 const browser = await puppeteer.launch({ args: ['--no-sandbox'] });  
 const page = await browser.newPage();  
 await page.setContent(html, { waitUntil: 'networkidle0' });  
 await page.pdf({ path: pdfOut, format: 'A4', printBackground: true });  
 await browser.close();  
}

## 12.8 TIDeS‑CHK Template & Mapping

**docs/templates/tides-chk.html.j2**

<section aria-label="TIDeS-CHK">  
 <h2>TIDeS‑CHK — 18‑Item Checklist</h2>  
 <ol>  
 <li>Spec version present (`spec-version`) — <strong>{{ yesno(checks.spec) }}</strong></li>  
 <li>Injection start recorded (`timing-injection`) — <strong>{{ yesno(checks.injection) }}</strong></li>  
 <li>UCUM units used (`unit-ucum`) — <strong>{{ yesno(checks.ucum) }}</strong></li>  
 <li>FoR present for dose (`registration-fuid`) — <strong>{{ yesno(checks.for) }}</strong></li>  
 <li>Provenance (software+hash) (`provenance`) — <strong>{{ yesno(checks.prov) }}</strong></li>  
 <li>Sampling adequacy checked (`sampling-adequacy`) — <strong>{{ yesno(checks.sampling) }}</strong></li>  
 <li>Safety policy declared (`policy-declared`) — <strong>{{ yesno(checks.policy) }}</strong></li>  
 <li>Kidney/liver/marrow constraints evaluated (`constraint-\*`) — <strong>{{ yesno(checks.constraints) }}</strong></li>  
 <li>FLASH coverage reported — <strong>{{ yesno(checks.flash) }}</strong></li>  
 <li>SEG labels coded (`seg-coded-meanings`) — <strong>{{ yesno(checks.seg) }}</strong></li>  
 <li>RTDOSE units Gy (`rtdose-for-match`) — <strong>{{ yesno(checks.rtdose) }}</strong></li>  
 <li>FHIR DiagnosticReport present — <strong>{{ yesno(checks.fhir) }}</strong></li>  
 <li>OMOP rows exported — <strong>{{ yesno(checks.omop) }}</strong></li>  
 <li>Uncertainty method documented — <strong>{{ yesno(checks.uncert) }}</strong></li>  
 <li>PK fit diagnostics recorded — <strong>{{ yesno(checks.pkdiag) }}</strong></li>  
 <li>Kernel provenance declared — <strong>{{ yesno(checks.kernel) }}</strong></li>  
 <li>Cumulative accounting considered — <strong>{{ yesno(checks.cumulative) }}</strong></li>  
 <li>Report signed and archived — <strong>{{ yesno(checks.archive) }}</strong></li>  
 </ol>  
</section>

**Utility (Jinja2 filter):**

env.filters['yesno'] = lambda b: 'Yes' if b else 'No'

## 12.9 Figures & DVH Rendering (Optional)

* DVH generation **SHOULD** use the voxel dose arrays (Ch.5) to compute cumulative histograms per ROI; export as inline SVG for crisp print.
* For reproducibility, seed plotting libraries and avoid random jitter; set fonts to system default.

**Python (matplotlib → SVG)**

import matplotlib.pyplot as plt  
import io, base64  
  
def dvh\_svg(dose\_vals, bins=256):  
 hist, edges = np.histogram(dose\_vals, bins=bins, range=(0, max(1e-6, dose\_vals.max())))  
 c = 1.0 - np.cumsum(hist[::-1]) / hist.sum()  
 x = edges[1:]  
 fig = plt.figure(figsize=(3,2.2), dpi=160)  
 plt.plot(x, c)  
 plt.xlabel('Dose (Gy)'); plt.ylabel('Volume frac')  
 buf = io.BytesIO(); plt.savefig(buf, format='svg', bbox\_inches='tight'); plt.close(fig)  
 return buf.getvalue().decode()

Embed with <figure><figcaption>DVH — Liver</figcaption>{{ dvh\_svg|safe }}</figure>.

## 12.10 FHIR Packaging of Reports

* Attach the PDF to a FHIR DiagnosticReport.presentedForm (application/pdf, base64).
* The HTML **MAY** also be attached as text/html.
* Use Binary resource with SHA‑256 metadata and link from DiagnosticReport.

**Example snippet (JSON):**

{  
 "resourceType":"DiagnosticReport",  
 "presentedForm":[{"contentType":"application/pdf","data":"<base64>","title":"TIDeS Clinical Report"}]  
}

## 12.11 Signing & Archival

* Compute SHA‑256 for HTML/PDF; store in a sidecar JSON (report.meta.json) with software/version.
* Optional Ed25519 signature (reuse Chapter 10 badge keys) over the hash payload; keep signatures alongside reports.

**report.meta.json**

{"sha256":"...","generator":"tides-reporter 1.0.0","issuedAt":"2025-09-25T12:44Z"}

## 12.12 CLI & API Contracts

### 12.12.1 CLI

$ tides report build --doc doc.json --clinical out/clinical.pdf --html out/clinical.html  
$ tides report build --doc doc.json --research out/research.pdf  
$ tides report chk --doc doc.json --out out/tides-chk.pdf  
$ tides report bundle --doc doc.json --fhir out/diagnosticreport.json --attach out/clinical.pdf

### 12.12.2 HTTP (extract)

post: /report/build  
requestBody:  
 content:  
 application/json:  
 schema: { $ref: 'https://tides.org/schemas/1.0.0/tides-reportdoc.schema.json' }  
responses:  
 '200': { description: PDF bytes (base64) }

## 12.13 Localization Packs

* i18n/en.yaml, i18n/fr.yaml, i18n/de.yaml, i18n/es.yaml with keys: title, subject, policy, validator, organSummary, provenance, flashCoverage, samplingAdequacy, statuses (PASS/WARN/ERROR).
* Renderer loads the pack matching doc.locale and substitutes labels; numeric values and UCUM units are not translated.

## 12.14 QA & Acceptance Tests

* Golden HTML/PDF byte checksums committed for the canonical fixtures (Chapter 11).
* Lighthouse accessibility audit score ≥ 90 for HTML.
* PDF/UA (tagged PDF) verification with veraPDF in CI.
* Diff tolerance: **zero** (exact match) except for timestamp fields.

**GitHub Actions (extract):**

- run: python tools/render\_all.py # renders clinical + research for all fixtures  
- run: sha256sum -c .golden/reports.sha256  
- run: verapdf out/\*.pdf --offersist

## 12.15 Troubleshooting

| Symptom | Cause | Fix |
| --- | --- | --- |
| PDF fonts look wrong | Missing fonts on runner | Embed system fonts or ship webfonts |
| Non‑deterministic charts | Timestamped seeds or jitter | Fix seeds; remove stochastic layout |
| Screen reader skips table headers | Missing scope="col" | Add scope attributes |

## 12.16 Security & Privacy

* **No PHI** in generated outputs by default; overrides require explicit flags.
* Links to validator reports should be local file URLs or object‑store with signed URLs; never public by default.
* Remove EXIF/metadata from embedded images before packaging.

## 12.17 Chapter Summary

* Clinical and research reports render deterministically from TIDeS artifacts; accessible, localized, and auditable.
* Templates and reference renderers (Python/Node) produce HTML/PDF with TIDeS‑CHK and validator embedding.
* Signing, hashing, and FHIR packaging close the loop for exchange and archival.

**End of Chapter 12 (Normative & Executable).**

# TIDeS Handbook — Chapter 13

## Site & Vendor Implementation Path (From Zero → A‑PASS)

**Purpose.** Provide a complete, field‑ready **playbook** for deploying TIDeS from scratch at a hospital, research lab, or vendor. You will prepare data sources, build the dosimetry pipeline, integrate DICOM/FHIR/OMOP, harden security, wire the validator and reports, pass conformance, and operate safely in production with monitoring and change control. This chapter includes end‑to‑end checklists, IaC deployment templates, orchestration DAGs, ETL, SOPs, and acceptance tests aligned to Chapters 1–12.

**Audience.** Enterprise imaging IT, clinical physics/NM teams, platform engineers, vendor implementers, QA, and auditors.

**Outcome.** A site or product moves from greenfield to **A‑PASS** (or B/C) with reproducible infrastructure and documentation.

**Normative keywords:** **MUST**, **SHOULD**, **MAY**.

## 13.0 Overview & Milestones

**M0 — Readiness (Week 0–2).** Roles/RACI defined, data inventory complete, security posture agreed.

**M1 — Data Conformance (Week 2–4).** DICOM export policy configured; SEG labels coded; RTDOSE lossless policy set.

**M2 — Pipeline Minimal (Week 3–6).** PK fit + TIA + voxel dose map; provenance hashing; fixtures validated.

**M3 — Interop (Week 5–7).** FHIR DiagnosticReport/Observations online; OMOP DDL applied; round‑trips pass.

**M4 — Validator & Safety (Week 6–8).** Policy pack loaded; TIDeS validator in CI; A‑PASS dry‑run.

**M5 — Reporting & Go‑Live (Week 8–10).** Clinical 2‑page report, research report; SOPs active; badge issued.

## 13.1 RACI (Roles & Responsibilities)

| Task | Physicist | NM Physician | PACS/VNA | Platform Eng | Data/Analytics | QA/Reg | Vendor |
| --- | --- | --- | --- | --- | --- | --- | --- |
| DICOM policy & tags | C | I | **R** | **A** | I | I | C |
| SEG coding policy | **A** | R | C | I | I | I | C |
| PK/dose pipeline | **R** | C | I | **A** | C | I | C |
| Validator/CI | C | I | I | **A** | C | **R** | C |
| FHIR/OMOP | I | C | I | **A** | **R** | C | C |
| Safety policy | **A** | **R** | I | I | I | C | C |
| Reports & SOPs | **A** | **R** | I | C | I | C | C |

R: Responsible, A: Accountable, C: Consulted, I: Informed.

## 13.2 Environment Reference Architectures

### 13.2.1 Single‑Node (PoC / Secure Lab)

* **Docker Compose** stack; offline by default.
* Components: tides-pipeline, tides-validator, tides-reporter, dicomweb-proxy, fhir-server, omop-db.

**deploy/compose.yml**

version: "3.9"  
services:  
 fhir:  
 image: hapi-fhir/hapi:v6  
 environment: { hapi.fhir.allow\_external\_references: "false" }  
 ports: ["8080:8080"]  
 omop:  
 image: postgres:15  
 environment:  
 POSTGRES\_DB: omop  
 POSTGRES\_USER: omop  
 POSTGRES\_PASSWORD: omop  
 volumes: ["omop-data:/var/lib/postgresql/data"]  
 dicomweb:  
 image: ohif/dicomweb-proxy:latest  
 environment: { DICOMWEB\_QIDO\_ROOT: "http://pacs:8042/dicom-web", DICOMWEB\_WADO\_ROOT: "http://pacs:8042/dicom-web" }  
 tides-pipeline:  
 build: ../pipeline  
 environment: { TIDES\_POLICY: "/policy/policy-oar-1.0.0.yaml" }  
 volumes: ["../policy:/policy", "../examples:/data"]  
 tides-validator:  
 build: ../validator  
 command: ["python","-m","validator.cli","validate","/data/bundle.json","--profile","A","--format","html"]  
 volumes: ["../examples/case\_pass\_minimal:/data"]  
 tides-reporter:  
 build: ../reporter  
 volumes: ["../out:/out"]  
volumes: { omop-data: {} }

### 13.2.2 HA Production (Kubernetes)

* Separate namespaces: tides-core, tides-sec.
* Use CSI encrypted volumes; NetworkPolicies default‑deny.

**deploy/k8s/tides-core.yaml (extract)**

apiVersion: apps/v1  
kind: Deployment  
metadata: { name: tides-validator, namespace: tides-core }  
spec:  
 replicas: 2  
 selector: { matchLabels: { app: tides-validator } }  
 template:  
 metadata: { labels: { app: tides-validator } }  
 spec:  
 containers:  
 - name: validator  
 image: ghcr.io/tides/validator:1.0.0  
 args: ["validate","/mnt/bundle.json","--profile","A","--format","json"]  
 volumeMounts: [{ name: work, mountPath: /mnt }]  
 volumes: [{ name: work, emptyDir: {} }]  
---  
apiVersion: networking.k8s.io/v1  
kind: NetworkPolicy  
metadata: { name: deny-egress, namespace: tides-core }  
spec:  
 podSelector: {}  
 policyTypes: [Egress]  
 egress: []

**deploy/k8s/values.yaml (Helm values)**

policyPacks:  
 - name: policy-oar  
 version: 1.0.0  
 configMap: tides-policy-oar  
security:  
 allowNet: false  
 redactLogs: true  
profiles: [A,B]

## 13.3 Data Conformance (Chapter 3 & 7 Alignment)

**Imaging export policy (PACS/VNA):** 1. Enhanced PET/NM/CT/MR only; disable classic legacy where possible.  
2. **Lossless** transfer syntaxes for dosimetry inputs.  
3. **SEG**: Segment labels **MUST** carry SNOMED/RadLex codes and algorithm provenance.  
4. **RTDOSE**: DoseUnits = GY; include FrameOfReferenceUID, GridFrameOffsetVector, scaling.  
5. **Spatial Registration** stored for any resampling.

**Checklist (site policy):** - [ ] DICOM conformance statement updated.  
- [ ] OHIF/Viewer shows coded ROIs.  
- [ ] Test fixture case\_registration\_ok parsed successfully.

## 13.4 Pipeline Assembly (PK → TIA → Dose → Report)

**Reference Makefile**

.PHONY: all fit dose report validate  
DATA?=examples/case\_pass\_minimal  
all: fit dose report validate  
fit:  
 python pipeline/fit\_pk.py $(DATA)/imaging.json > $(DATA)/pkmodel.json  
TIA:  
 python pipeline/tia\_integrate.py $(DATA)/times.json > $(DATA)/tia.json  
dose:  
 python pipeline/convolve.py $(DATA)/pkmodel.json $(DATA)/kernel.json > $(DATA)/dosemap.json  
report:  
 python pipeline/summarize.py $(DATA)/dosemap.json $(DATA)/seg.json > $(DATA)/dosereport.json  
validate:  
 python -m validator.cli validate $(DATA)/bundle.json --profile A --format html > $(DATA)/report.html

**Airflow DAG (optional)**

from airflow import DAG  
from airflow.operators.bash import BashOperator  
from datetime import datetime  
with DAG('tides\_pipeline', start\_date=datetime(2025,1,1), schedule\_interval=None, catchup=False) as dag:  
 fit = BashOperator(task\_id='fit', bash\_command='python pipeline/fit\_pk.py {{ dag\_run.conf["imaging"] }} > /tmp/pk.json')  
 dose = BashOperator(task\_id='dose', bash\_command='python pipeline/convolve.py /tmp/pk.json {{ dag\_run.conf["kernel"] }} > /tmp/dose.json')  
 report = BashOperator(task\_id='report', bash\_command='python pipeline/summarize.py /tmp/dose.json {{ dag\_run.conf["seg"] }} > /tmp/report.json')  
 validate = BashOperator(task\_id='validate', bash\_command='python -m validator.cli validate {{ dag\_run.conf["bundle"] }} --profile A --format json > /tmp/val.json')  
 fit >> dose >> report >> validate

## 13.5 FHIR & OMOP Integration

**FHIR submission script (integrations/fhir\_push.py)**

import requests, json, sys  
FHIR=f"{sys.argv[1]}"; BUNDLE=sys.argv[2]  
B=json.load(open(BUNDLE))  
# Convert DoseReport → FHIR Bundle (Chapter 7 helper)  
from interop.fhir import tides\_to\_fhir\_bundle  
bundle = tides\_to\_fhir\_bundle(B['resources'][-1])  
r = requests.post(f"{FHIR}/Bundle", json=bundle, timeout=10)  
r.raise\_for\_status(); print('FHIR OK', r.status\_code)

**OMOP ETL (integrations/omop\_etl.sql)**

INSERT INTO tides\_study(study\_uid, person\_id, nuclide, start\_ts)  
SELECT :study\_uid, :person\_id, :nuclide, :start\_ts;  
INSERT INTO tides\_roi\_dose(study\_uid, roi\_code, roi\_text, dmean\_gy, dmax\_gy, uncertainty\_pc)  
VALUES (:study\_uid, :roi\_code, :roi\_text, :dmean, :dmax, :unc);

**Validation:** run validator link-referential, fhir-profile-declared, omop-ddl-version post‑ETL.

## 13.6 Safety Policy Wiring (Chapter 6)

* Load policy-oar-1.0.0.yaml; confirm **jurisdiction tags**.
* Configure pediatric modifiers if relevant.
* Ensure cumulative accounting across repeated TidesStudy IDs.

**Policy injection in pipeline (pipeline/policy\_eval.py)**

import yaml, json, sys  
pol = yaml.safe\_load(open(sys.argv[1]))  
rep = json.load(open(sys.argv[2]))  
# Evaluate simple Dmean thresholds per ROI (see Chapter 6 engine for full logic)  
findings=[]  
limits={ (o['roiCode'],o['roiCodeText']): (o['limit'], o['warnThreshold']) for o in pol['oars'] }  
for s in rep['series']:  
 for o in s['organSummaries']:  
 lim, warn = limits.get((o['roiCode'], o.get('roiCodeText','')), (None,None))  
 if lim is None: continue  
 dec = 'PASS'  
 if o['Dmean\_Gy'] >= lim: dec='ERROR'  
 elif o['Dmean\_Gy'] >= warn: dec='WARN'  
 findings.append({ 'roiCode': o['roiCode'], 'decision': dec })  
json.dump({'policy': {'name': pol['name'], 'version': pol['version']}, 'findings': findings}, sys.stdout)

## 13.7 Security & Privacy Hardening (Chapter 15 hooks)

**Controls (MUST for Profile A production):** - Network egress disabled on validator/pipeline pods; no PHI in logs.  
- All artifacts PHI‑free; only **URNs** persisted; display identifiers injected at render time only if approved.  
- Disk encryption for OMOP/Postgres; backups encrypted; keys in HSM/KMS.  
- Audit trails: who ran which pipeline and validator version + hash.

**OPA Gate (deploy/opa/policy.rego)**

package tides  
  
default allow = false  
allow {  
 input.request.kind.kind == "Deployment"  
 not input.request.object.spec.template.spec.containers[\_].env[\_].name == "ALLOW\_NET"  
}

## 13.8 Observability (Logs, Metrics, Traces)

* **Metrics:** pipeline durations, voxel throughput, dose calc cache hit‑rate, validator rule counts, badge outcomes.
* **Prometheus exporter (monitoring/metrics.py)**

from prometheus\_client import Counter, Gauge, start\_http\_server  
VALIDATIONS = Counter('tides\_validations\_total','count', ['profile','status'])  
RULES = Gauge('tides\_rule\_failures','failures per rule', ['ruleId'])  
start\_http\_server(9108)  
# elsewhere after run  
# VALIDATIONS.labels('A','PASS').inc()  
# for f in findings: RULES.labels(f.ruleId).inc()

* **Dashboards:** Grafana panels for PASS% by month, WARN distribution, top failing rules.

## 13.9 CI/CD & Gates

**GitHub Actions (.github/workflows/site.yml)**

name: Site CI  
on: [push]  
jobs:  
 validate:  
 runs-on: ubuntu-latest  
 steps:  
 - uses: actions/checkout@v4  
 - uses: actions/setup-python@v5  
 with: { python-version: '3.11' }  
 - run: pip install -r validator/requirements.txt  
 - run: python tools/validate\_all.py  
 badge:  
 needs: validate  
 runs-on: ubuntu-latest  
 steps:  
 - uses: actions/checkout@v4  
 - run: python tools/make\_badge.py examples/case\_pass\_minimal/bundle.json > .out/badge.json

**Gates:** merge blocked unless fixtures match goldens and A-PASS badge is produced for site reference bundle.

## 13.10 Operational SOPs (Templates)

**SOP‑01 — Daily Pipeline Check** - [ ] Validate previous day’s studies; investigate any ERRORs.  
- [ ] Review WARNs near limits; consult physician if clinical impact.  
- [ ] Backup OMOP DB; verify checksum.

**SOP‑02 — New Nuclide Onboarding** - [ ] Kernel provenance documented; grid compatibility tested.  
- [ ] Update policy file if OAR references differ.  
- [ ] Add fixture; update goldens; run connectathon test.

**SOP‑03 — Release & Change Control** - [ ] Rule pack SemVer bump; DOI minted.  
- [ ] Regenerate badges for fixtures; archive.

## 13.11 Risk Register (Starter)

| Risk | Likelihood | Impact | Mitigation |
| --- | --- | --- | --- |
| Wrong UCUM codes | M | H | Validator unit-ucum ERROR; fixtures cover case 8 |
| Missing FoR link | L | H | Rule registration-fuid; CI gate |
| PHI leakage in logs | L | H | Redaction on reporters; OPA gate; log review SOP |
| Kernel/grid mismatch | M | M | Conformance tests; Chapter 5 metadata checks |

## 13.12 Migration (Legacy → TIDeS)

1. **Inventory legacy organ‑level CSVs**; map to TidesDoseReport (§8.7).
2. **Optional** voxel back‑fill via RTDOSE if available.
3. Start on **Profile C** for retrospective datasets; graduate to **B/A** as voxel/interop added.

**CSV → TIDeS (migrations/csv\_to\_tides.py)**

import csv, json, sys  
out = {"specVersion":"1.0.0","id":"urn:tides:bundle:mig","study":"urn:tides:study:mig","resources":[],"provenance":{"software":"csv-migr","version":"1.0.0","hash":"N/A","inputs":["csv"]}}  
series={"doseMap":"urn:tides:dosemap:legacy","organSummaries":[]}  
for row in csv.DictReader(open(sys.argv[1])):  
 series['organSummaries'].append({"roiCode":row['roi\_code'],"roiCodeText":row['roi\_text'],"Dmean\_Gy":float(row['dmean\_gy']),"Dmax\_Gy":float(row['dmax\_gy']),"uncertainty\_pc":float(row.get('uncertainty\_pc',0))})  
out['resources'].append({"resourceType":"TidesDoseReport","id":"urn:tides:dosereport:mig","subject":"urn:tides:subject:mig","series":[series]})  
json.dump(out, open(sys.argv[2],'w'), indent=2)

## 13.13 Training & Enablement

* **Play sessions:** run fixtures #1–#10 start‑to‑finish; review validator reports and TIDeS‑CHK.
* **Shadowing:** physicist pairs with platform engineer for pipeline steps.
* **Knowledge base:** site wiki linking spec clauses to SOPs.

## 13.14 Acceptance Tests (Go‑Live)

**AT‑1:** Process 3 synthetic studies end‑to‑end → **A‑PASS** badge; HTML/PDF reports generated and archived.  
**AT‑2:** Inject case\_safety\_violation\_kidney → validator **ERROR**; alert to clinician per SOP.  
**AT‑3:** FHIR push round‑trip → resources retrievable and profile‑valid.  
**AT‑4:** Disaster recovery restore of OMOP DB within RTO/RPO targets.

## 13.15 Post‑Go‑Live Monitoring & KPIs

* **Conformance rate (A‑PASS%)** monthly ≥ 95%.
* **Median pipeline runtime** ≤ N minutes per study (site‑defined).
* **WARN incidence** trend; top 3 rule offenders.
* **Report turnaround** from imaging complete → PDF delivered.

## 13.16 Vendor Productization Notes

* Ship SBOM (validator/pipeline/reporters).
* Expose **read‑only** Validator API; no PHI.
* Provide capability.json and sample **A‑PASS** evidence bundle to prospects.
* Support site policy overlays and waiver workflows (Chapter 10).

## 13.17 Checklists (Printable)

**Site Readiness** - [ ] Data inventory complete; DICOM policy set; SEG codes cataloged.  
- [ ] OMOP DDL applied; FHIR endpoint reachable.  
- [ ] Security baseline: network egress blocked; encryption on.  
- [ ] Staff trained; SOPs approved.

**Go‑Live** - [ ] A‑PASS achieved on reference case.  
- [ ] Clinical report template reviewed by NM physician.  
- [ ] Monitoring dashboards live; alert routes tested.  
- [ ] Backup/DR tested.

## 13.18 Chapter Summary

* A concrete, reproducible path from zero to **A‑PASS**: data conformance → pipeline → interop → validator → safety → reporting → operations.
* IaC, DAGs, ETL, security/observability, SOPs, and acceptance tests ensure a safe, auditable deployment that honors TIDeS’ normative requirements.

**End of Chapter 13 (Operational & Executable).**

# TIDeS Handbook — Chapter 14

## Repository Layout (Reference Monorepo, Tooling, and Release Ops)

**Purpose.** Define the **canonical** repository structure and all developer‑experience (DX) assets required to build, validate, release, and operate TIDeS artifacts. Includes file tree, scaffolding scripts, coding standards, CI/CD, docs site wiring, security posture, contribution flow, and release/DOI process. All files here are **executable templates**—copy into your repo as‑is, then tailor.

**Audience.** Maintainers, vendors, site integrators, contributors, DevOps.

**Outcome.** You will have a production‑grade monorepo that reproducibly builds schemas, validator, fixtures, reports, docs, and packages; enforces style and licenses; and publishes signed releases with DOIs and badges.

**Normative keywords:** **MUST**, **SHOULD**, **MAY**.

## 14.0 Top‑Level Tree (Authoritative)

TIDES/  
 spec/ # Chapters 0–18 (handbook / IG)  
 schemas/ # JSON Schema 2020‑12 (Ch.8)  
 validator/ # Reference validator (Ch.9)  
 examples/ # Canonical fixtures (Ch.11)  
 reporter/ # HTML/PDF renderers & templates (Ch.12)  
 interops/ # DICOM/FHIR/OMOP mappings (Ch.7)  
 omop/ # DDL + ETL helpers (Ch.7,13)  
 profiles/ # A/B/C profile overlays (Ch.10)  
 pipeline/ # Reference PK/TIA/Dose pipeline stubs (Ch.5)  
 deploy/ # Compose & Kubernetes (Ch.13)  
 docs/ # IG site, MkDocs or Docusaurus  
 tools/ # Generators, CLIs, misc (Ch.11)  
 .github/ # Workflows, issue/PR templates  
 .devcontainer/ # VS Code Dev Containers  
 .vscode/ # Tasks & settings  
 .licenses/ # Third‑party notices  
 LICENSE # Apache‑2.0 (code)  
 LICENSE-text/CC-BY-4.0.txt# CC‑BY‑4.0 (spec)  
 CITATION.cff # Citation metadata (DOI)  
 CODE\_OF\_CONDUCT.md # Contributor Covenant  
 CONTRIBUTING.md # How to contribute  
 SECURITY.md # Vulnerability disclosure  
 GOVERNANCE.md # Maintainer/RFC process  
 RELEASE.md # Release/DOI playbook  
 CHANGELOG.md # Auto‑generated via semantic‑release  
 CODEOWNERS # Review paths → owners  
 .gitignore # Ignore sets  
 .gitattributes # LF endings, linguist, export‑ignore  
 pyproject.toml # Python toolchain config (ruff/black/mypy)  
 package.json # Node toolchain for validator UI/reporters  
 Makefile # One‑command build/verify

**Normative:** All paths shown above **MUST** exist in reference distribution; consumers MAY merge or split but MUST keep $id URLs stable for schemas and rule packs.

## 14.1 Scaffolding Scripts (Bootstrap in 60s)

### 14.1.1 tools/bootstrap.sh

#!/usr/bin/env bash  
set -euo pipefail  
mkdir -p schemas/1.0.0 validator/rules examples docs/templates profiles deploy .github/workflows  
mkdir -p reporter templates tools/gen omop interops/dicom interops/fhir interops/omop  
cp -n LICENSE LICENSE-text/CC-BY-4.0.txt || true  
cat > .gitignore <<'EOF'  
/.out/  
/.venv/  
/node\_modules/  
\_\_pycache\_\_/  
\*.pyc  
\*.DS\_Store  
\*.log  
\*.cache/  
/.pytest\_cache/  
/.coverage  
/dist/  
/site/  
EOF  
cat > CODEOWNERS <<'EOF'  
# Path owners  
/schemas/ @tides-core  
/validator/ @tides-core @qa-team  
/docs/ @docs-wg  
EOF

### 14.1.2 tools/check\_repo.py (Structure Linter)

import sys, os, json  
REQUIRED = [  
 'spec','schemas','validator','examples','reporter','interops','omop','profiles','pipeline','deploy','docs','.github']  
missing = [p for p in REQUIRED if not os.path.isdir(p)]  
print(json.dumps({"missing": missing}, indent=2))  
sys.exit(1 if missing else 0)

## 14.2 Coding Standards & Linters

### 14.2.1 pyproject.toml

[tool.black]  
line-length = 100  
[tool.ruff]  
line-length = 100  
select = ["E","F","I","B","UP","ANN"]  
ignore = ["ANN101","ANN102"]  
[tool.mypy]  
python\_version = "3.11"  
strict = true  
warn\_unused\_ignores = true  
[tool.isort]  
profile = "black"

### 14.2.2 package.json (Node lint/test)

{  
 "name": "tides-monorepo",  
 "private": true,  
 "scripts": {  
 "lint:js": "eslint . --ext .js,.mjs,.ts",  
 "lint": "npm run lint:js && ruff check .",  
 "test": "vitest run",  
 "build": "echo 'schemas & docs build in workflows'"  
 },  
 "devDependencies": {  
 "eslint": "^8",  
 "@typescript-eslint/parser": "^7",  
 "@typescript-eslint/eslint-plugin": "^7",  
 "vitest": "^1",  
 "typescript": "^5"  
 }  
}

### 14.2.3 .pre-commit-config.yaml

repos:  
 - repo: https://github.com/psf/black  
 rev: 24.3.0  
 hooks: [{id: black}]  
 - repo: https://github.com/astral-sh/ruff-pre-commit  
 rev: v0.6.5  
 hooks: [{id: ruff}]  
 - repo: https://github.com/pre-commit/mirrors-prettier  
 rev: v3.3.3  
 hooks: [{id: prettier}]  
 - repo: https://github.com/shellcheck-py/shellcheck-py  
 rev: v0.10.0.1  
 hooks: [{id: shellcheck}]  
 - repo: local  
 hooks:  
 - id: repo-structure  
 name: repo-structure  
 entry: python tools/check\_repo.py  
 language: system  
 pass\_filenames: false

## 14.3 CI/CD Workflows (GitHub Actions)

### 14.3.1 .github/workflows/validate.yml

name: Validate Schemas & Fixtures  
on: [push, pull\_request]  
permissions: { contents: read }  
jobs:  
 lint-and-validate:  
 runs-on: ubuntu-latest  
 steps:  
 - uses: actions/checkout@v4  
 - uses: actions/setup-python@v5  
 with: { python-version: '3.11' }  
 - run: pip install -r validator/requirements.txt  
 - run: pip install pre-commit jinja2 weasyprint  
 - run: pre-commit run --all-files  
 - run: python tools/validate\_all.py  
 - name: Render Reports (canonical)  
 run: python tools/render\_all.py  
 - name: Upload HTML/PDF artifacts  
 uses: actions/upload-artifact@v4  
 with: { name: tides-reports, path: out/ }

### 14.3.2 .github/workflows/release.yml

name: Release & DOI  
on:  
 push:  
 tags:  
 - 'v\*.\*.\*'  
jobs:  
 build-publish:  
 runs-on: ubuntu-latest  
 steps:  
 - uses: actions/checkout@v4  
 - name: Create Release Assets  
 run: |  
 tar czf tides-schemas-${{ github.ref\_name }}.tar.gz schemas  
 tar czf tides-validator-${{ github.ref\_name }}.tar.gz validator  
 sha256sum \*.tar.gz > SHASUMS256.txt  
 - uses: softprops/action-gh-release@v2  
 with:  
 files: |  
 tides-schemas-${{ github.ref\_name }}.tar.gz  
 tides-validator-${{ github.ref\_name }}.tar.gz  
 SHASUMS256.txt  
 zenodo-doi:  
 needs: build-publish  
 runs-on: ubuntu-latest  
 steps:  
 - uses: actions/checkout@v4  
 - name: Prepare Zenodo metadata  
 run: python tools/doi\_publish.py --tag ${{ github.ref\_name }} --out zenodo.json  
 - name: Upload to Zenodo (placeholder)  
 run: echo 'POST zenodo.json (site‑specific secret)'

### 14.3.3 .github/workflows/security.yml

name: Security Checks  
on: [push, pull\_request, schedule]  
jobs:  
 codeql:  
 uses: github/codeql-action/.github/workflows/codeql.yml@v3  
 trivy:  
 runs-on: ubuntu-latest  
 steps:  
 - uses: actions/checkout@v4  
 - uses: aquasecurity/trivy-action@0.24.0  
 with: { scan-type: fs, ignore-unfixed: true }

## 14.4 Documentation Site (MkDocs)

### 14.4.1 docs/mkdocs.yml

site\_name: TIDeS 1.0.0 — Implementation Guide  
repo\_url: https://github.com/tides/tides  
theme:  
 name: material  
 features: [navigation.sections, toc.integrate, search.suggest]  
nav:  
 - Home: index.md  
 - Spec:  
 - Chapter 0: spec/00-governance.md  
 - Chapter 1: spec/01-axioms.md  
 - ...  
 - Schemas: schemas/index.md  
 - Validator: validator/index.md  
 - Fixtures: examples/index.md  
 - Reporting: reporter/index.md  
markdown\_extensions: [admonition, toc, tables]  
plugins: [search]

### 14.4.2 docs/index.md

# TIDeS — Theranostics Interoperability, Dosimetry & Safety  
  
Welcome to the TIDeS Implementation Guide. Start with \*\*Chapter 0\*\* and follow the left‑nav. For runnable artifacts, see the \*\*Schemas\*\*, \*\*Validator\*\*, and \*\*Examples\*\* sections.

### 14.4.3 docs/Makefile

serve:  
 mkdocs serve -a 0.0.0.0:8000  
build:  
 mkdocs build --strict

## 14.5 Legal & Metadata

### 14.5.1 LICENSE (Apache‑2.0)

* Single file applies to **code** ( validator, pipeline, tools, reporter).

### 14.5.2 LICENSE-text/CC-BY-4.0.txt

* Applies to **text/spec** (spec/, docs/). Link from README.

### 14.5.3 CITATION.cff

cff-version: 1.2.0  
title: TIDeS — Theranostics Interoperability, Dosimetry & Safety  
version: 1.0.0  
doi: 10.5281/zenodo.TBD  
authors:  
 - family-names: Core  
 given-names: TIDeS Working Group  
license: Apache-2.0

### 14.5.4 SECURITY.md

# Security Policy  
  
Report vulnerabilities to security@tides.org. Do not open public issues for sensitive findings. We commit to triage within 5 business days.

### 14.5.5 GOVERNANCE.md (RFC/balloting)

# Governance & RFCs  
  
Changes follow the RFC process (Chapter 16). Proposals are numbered RFC‑YYYY‑NN. Maintainers ballot changes; SemVer bumps upon merge.

## 14.6 Issue & PR Templates

### 14.6.1 .github/ISSUE\_TEMPLATE/bug\_report.md

---  
name: Bug report  
about: Report a problem in the TIDeS repo  
---  
\*\*Describe the bug\*\*  
\*\*To Reproduce\*\*  
\*\*Expected behavior\*\*  
\*\*Artifacts\*\* (bundle.json, report.json)  
\*\*Environment\*\* (validator version)

### 14.6.2 .github/ISSUE\_TEMPLATE/feature\_request.md

---  
name: Feature request  
about: Suggest an idea for TIDeS  
---  
\*\*Problem\*\*  
\*\*Proposal\*\* (link to spec clause)  
\*\*Impact\*\*

### 14.6.3 .github/PULL\_REQUEST\_TEMPLATE.md

## Summary  
  
## Type of change  
- [ ] Spec text  
- [ ] Schema  
- [ ] Validator rule  
- [ ] Fixture  
- [ ] Docs/Reporting  
  
## Checklist  
- [ ] Updated CHANGELOG.md  
- [ ] Added/updated tests/fixtures  
- [ ] SemVer bump if breaking

## 14.7 Dev Environments

### 14.7.1 .devcontainer/devcontainer.json

{  
 "name": "TIDeS Dev",  
 "image": "mcr.microsoft.com/devcontainers/python:3.11",  
 "features": { "ghcr.io/devcontainers/features/node:1": {} },  
 "postCreateCommand": "pip install -r validator/requirements.txt && npm i",  
 "customizations": {  
 "vscode": {  
 "extensions": ["ms-python.python","esbenp.prettier-vscode","charliermarsh.ruff"]  
 }  
 }  
}

### 14.7.2 .vscode/settings.json

{ "editor.formatOnSave": true, "files.eol": "\n" }

## 14.8 Makefile (Single‑entry DX)

.PHONY: all clean lint test validate docs release  
all: lint test validate docs  
lint:  
 pre-commit run --all-files || (echo "Run 'pre-commit install'" && exit 1)  
test:  
 pytest -q || true # placeholder until tests added  
validate:  
 python tools/validate\_all.py  
docs:  
 $(MAKE) -C docs build  
release:  
 @echo "Tag vX.Y.Z and push; GH Action will publish artifacts & DOI"  
clean:  
 rm -rf .out out site .pytest\_cache

## 14.9 Packaging & Distribution

* **Schema bundle**: tides-schemas-vX.Y.Z.tar.gz with $id‑stable layout.
* **Validator wheel**: pip wheel validator/ (optional PyPI).
* **Docker images**: ghcr.io/tides/validator:<tag>, ghcr.io/tides/reporter:<tag>.
* **NPM package** (optional): @tides/validator-core for Node consumers.

**validator/Dockerfile**

FROM python:3.11-slim  
WORKDIR /app  
COPY validator /app/validator  
RUN pip install -r /app/validator/requirements.txt  
ENTRYPOINT ["python","-m","validator.cli","validate"]

## 14.10 Git Hygiene

### 14.10.1 .gitattributes

\* text=auto eol=lf  
schemas/\* linguist-generated=true  
\*.pdf binary  
\*.tar.gz binary  
/docs/\*\* export-ignore  
/examples/\*\* export-ignore

### 14.10.2 .gitignore (expanded)

/.out/  
/out/  
/site/  
/node\_modules/  
/.venv/  
\_\_pycache\_\_/  
\*.pyc  
\*.coverage  
\*.log  
\*.svg

## 14.11 CHANGELOG & Semantic Release

* Conventional commits enforced (feat:, fix:, docs:, chore:).
* Release notes auto‑generated.

**CHANGELOG.md (excerpt template)**

# 1.0.0 (2025‑09‑25)  
  
### Features  
- Initial executable IG: schemas, validator, fixtures, reports.  
  
### Fixes  
- Tightened UCUM unit enums; corrected FoR regex.

## 14.12 Contributor Guide

**CONTRIBUTING.md**

## How to contribute  
1. Fork and create a feature branch.  
2. Run `make lint validate` locally; ensure fixtures pass.  
3. Open a PR referencing spec clauses & rule IDs.  
4. Maintainers review; squashed merge with Conventional Commit title.

**DCO (Developer Certificate of Origin)** — optional Signed-off-by: footer enforcement in CI.

## 14.13 Release/DOI Playbook

**RELEASE.md**

1. Update versions: `specVersion`, schemas `$id`, rule pack `version`.  
2. Regenerate goldens (Chapter 11) and reports (Chapter 12).  
3. Tag: `git tag vX.Y.Z && git push origin vX.Y.Z`.  
4. GitHub Action publishes tarballs; upload to Zenodo; copy DOI into `CITATION.cff`.  
5. Announce with profile badge criteria and upgrade notes.

## 14.14 Security, SBOM & Third‑Party Notices

* Generate SBOMs for Docker images (syft ghcr.io/tides/validator:tag).
* Store in .licenses/ with license summaries and notice files.
* Run vulnerability scans in security.yml workflow.

## 14.15 Repo Badges (README)

Add shields to README.md:

[![Validate](https://github.com/tides/tides/actions/workflows/validate.yml/badge.svg)]()  
[![Release](https://github.com/tides/tides/actions/workflows/release.yml/badge.svg)]()  
[![CodeQL](https://github.com/tides/tides/actions/workflows/security.yml/badge.svg)]()

## 14.16 Minimal README.md

# TIDeS — Theranostics Interoperability, Dosimetry & Safety (v1.0.0)  
  
This is the \*\*reference, executable\*\* Implementation Guide. Start with `/spec`, validate fixtures under `/examples`, and run the validator under `/validator`.  
  
## Quick start  
```bash  
make lint validate

## License

* Code: Apache‑2.0
* Text/spec: CC‑BY‑4.0 ```

## 14.17 Chapter Summary

* A prescriptive monorepo with scaffolding, lint/test/validate workflows, docs site, legal metadata, security scanning, packaging, and release/DOI automation.
* Copy these templates to stand up a compliant TIDeS repository with **minimal ceremony** and **maximal reproducibility**.

**End of Chapter 14 (Operational, Normative & Executable).**

# TIDeS Handbook — Chapter 15

## Security & Privacy (PHI‑Safe by Design)

**Purpose.** Define the **complete** security and privacy program for TIDeS—governing data protection from acquisition to archival across clinical and research settings. This chapter is normative and executable: it includes policies, controls, threat models, implementation code, audit schemas, incident playbooks, and compliance mappings (GDPR/HIPAA). All guidance aligns with Chapters 0–14 and is profile‑aware (A/B/C).

**Audience.** CISOs, privacy officers, platform engineers, clinical IT, vendors, auditors, and researchers.

**Outcome.** You will deploy TIDeS with **least privilege, zero egress by default, cryptographically provable integrity**, and privacy‑by‑design guardrails. You can prove compliance and pass audits **without exposing PHI**.

**Normative keywords:** **MUST**, **SHOULD**, **MAY**.

## 15.0 Principles (Normative)

1. **PHI‑minimalism.** Store **no PHI** in core artifacts (Bundles, DoseReports, rule packs, reports). Use URNs and unlinkable linkage keys held separately.
2. **Offline by default.** Validator, reporter, and pipeline **MUST** run with outbound network disabled unless explicitly allowed.
3. **Deterministic & auditable.** Every run emits a cryptographically signed, immutable audit log (hashes, versions, user, policy).
4. **Separation of duties.** Clinical identifiers and analysis artifacts are managed by different security principals and stores.
5. **Defense in depth.** Layered controls across identity, network, data, application, supply chain, and operations.
6. **Privacy by design.** Data minimization, purpose limitation, explicit legal basis/consent where required, and reproducible de‑identification.

## 15.1 Data Model: Identities & PHI Boundaries

### 15.1.1 Identifiers (Normative)

* **Core artifacts** MUST use **URNs**: urn:tides:subject:<UUID>, urn:tides:study:<UUID>.
* **Linkage keys** (mapping URN ↔ MRN/DOB/name) MUST be stored in a **separate, access‑controlled KVS** (e.g., HSM‑sealed Vault) with audit.
* **DICOM/FHIR** ingress MAY contain PHI, but downstream TIDeS exports and validator inputs **MUST** be PHI‑free.

### 15.1.2 PHI Surface Matrix

| Layer | Contains PHI? | Notes |
| --- | --- | --- |
| TIDeS Bundle / DoseReport | **No** | URNs only; clinical display IDs injected at render time via overlay |
| Validator Reports (JSON/TXT/HTML/SARIF) | **No** | Pointers only; no raw values except numeric metrics |
| DICOM inbound (PACS/VNA) | **Yes** | Must be isolated; de‑ID pipeline before TIDeS ingestion |
| FHIR server | **Yes/No** | Prefer pseudonymized subject references or consented zones |
| OMOP | **No** (preferred) | Store URNs; optional limited dataset with site policy |

## 15.2 Threat Modeling

* **Frameworks:** STRIDE (security) + LINDDUN (privacy).
* **Assets:** voxel dose maps, policy files, validator rule packs, signed badges, linkage KVS, DICOM ingress, FHIR endpoint, OMOP DB, CI secrets.
* **Adversaries:** insider misuse, ransomware, supply‑chain compromise, misconfigured cloud storage, data exfil via plugins.

**Canonical risks & controls**

| Risk | Vector | Control (Normative) |
| --- | --- | --- |
| PHI leakage in logs | verbose exception printing | Redaction middleware; logging policy forbids PHI; CI tests grep for PHI patterns |
| Rule‑pack tamper | altered severities | Signed rule packs; hash pinning; reproducible builds |
| Exfiltration | implicit network calls | --allow-net flag disabled; Kubernetes NetworkPolicy **deny‑all** egress |
| Supply chain | dep trojan | SBOM + sigstore verification; pinned hashes; offline cache |
| Linkage compromise | KVS exposure | HSM‑sealed Vault; per‑request short‑lived tokens; dual control to export |

## 15.3 Identity & Access Management (IAM)

### 15.3.1 Roles

* **viewer:** read‑only reports and badges.
* **validator:** run validator/reporters on PHI‑free bundles.
* **ingest:** handle DICOM/FHIR with PHI; cannot access validator outputs.
* **privacy‑officer:** approve de‑ID and linkage operations.
* **admin:** manage policy packs and rule packs; **cannot** read linkage KVS.

### 15.3.2 Authentication

* mTLS between services; OIDC for UI/API (Keycloak/Auth0).
* Service accounts use short‑lived JWTs (≤15 min) with audience restriction.

**Keycloak realm (excerpt)**

{  
 "realm": "tides",  
 "roles": {"realm": [{"name":"viewer"},{"name":"validator"},{"name":"ingest"},{"name":"privacy-officer"},{"name":"admin"}]},  
 "clients": [{"clientId":"tides-ui","publicClient":true,"redirectUris":["https://tides.example/ui/\*"],"defaultClientScopes":["profile","email"]}]  
}

### 15.3.3 Authorization (OPA/Rego)

package tides.auth  
  
default allow = false  
allow {  
 input.user.roles[\_] == "validator"  
 input.request.path == "/validate"  
}  
allow {  
 input.user.roles[\_] == "viewer"  
 input.request.path == "/reports"  
}  
deny\_linkage {  
 input.request.path == "/linkage"  
 not (input.user.roles[\_] == "privacy-officer")  
}

## 15.4 Network & Platform Security

* **Zero trust:** deny‑all egress; allowlist only FHIR/OMOP endpoints as needed.
* **TLS everywhere:** TLS 1.2+ with modern ciphers; certificates rotated automatically.
* **Isolation:** separate namespaces (tides-core, tides-sec); no node‑sharing with general workloads.

**Kubernetes NetworkPolicy (deny egress)**

apiVersion: networking.k8s.io/v1  
kind: NetworkPolicy  
metadata: { name: deny-egress, namespace: tides-core }  
spec:  
 podSelector: {}  
 policyTypes: [Egress]  
 egress: []

**Nginx mTLS (ingress)**

server {  
 listen 443 ssl;  
 ssl\_verify\_client on;  
 ssl\_client\_certificate /etc/nginx/ca.crt;  
 location /validate { proxy\_pass http://validator:8080; }  
}

## 15.5 Data Protection: Encryption & Keys

### 15.5.1 Encryption Policy (Normative)

* **At rest:** AES‑256‑GCM (storage class encryption) for volumes; per‑object envelope encryption for artifacts.
* **In transit:** TLS 1.2+ (mTLS for service‑to‑service).
* **Keys:** managed by KMS/HSM; rotation ≤ 365 days; dual‑control for export.

### 15.5.2 Envelope Encryption (Python)

from cryptography.hazmat.primitives.ciphers.aead import AESGCM  
from secrets import token\_bytes  
import json, base64  
  
def seal(obj: dict, kek: bytes) -> dict:  
 pt = json.dumps(obj, separators=(',',':')).encode()  
 dek = token\_bytes(32)  
 aes = AESGCM(dek)  
 nonce = token\_bytes(12)  
 ct = aes.encrypt(nonce, pt, None)  
 # Simulate KEK wrap (call KMS in production)  
 wrapped = AESGCM(kek).encrypt(b'0'\*12, dek, None)  
 return {"nonce": base64.b64encode(nonce).decode(), "ct": base64.b64encode(ct).decode(), "wrapped": base64.b64encode(wrapped).decode()}

## 15.6 De‑Identification (DICOM & FHIR)

### 15.6.1 DICOM De‑ID Policy (Profile A default)

* Remove/blank direct identifiers (0010,0010 PatientName, etc.).
* **Keep** technical tags needed for dosimetry (FoR, acquisition times, calibration factors).
* Assign new UID root for exported objects; map original↔new UIDs in linkage KVS only.

**DICOM tag table (excerpt)**

| Tag | Action | Rationale |
| --- | --- | --- |
| (0010,0010) PatientName | remove | PHI |
| (0010,0020) PatientID | remove | PHI |
| (0020,0052) FrameOfReferenceUID | **keep** | spatial consistency |
| (0054,1322) DecayCorrection | keep | dosimetry |

**pydicom de‑ID snippet**

import pydicom  
REMOVE = [(0x0010,0x0010),(0x0010,0x0020)]  
KEEP = [(0x0020,0x0052),(0x0054,0x1322)]  
  
def deid(ds):  
 for t in REMOVE:  
 if t in ds: del ds[t]  
 return ds

### 15.6.2 FHIR Pseudonymization

* Replace Patient.identifier with URN; move MRN to secure store; maintain link internally if legally required.
* Use **SMART scopes**; restrict to patient/DiagnosticReport.read where possible.

**FHIR example**

{"resourceType":"Patient","id":"urn:tides:subject:9a2f-...","identifier":[{"system":"urn:tides","value":"urn:tides:subject:9a2f-..."}]}

## 15.7 Logging, Auditing & Provenance

### 15.7.1 Audit Event Schema (JSON)

{  
 "ts":"2025-09-25T12:34:56Z",  
 "actor":"svc:validator",  
 "action":"validate",  
 "inputs":["sha256:...bundle"],  
 "software":{"name":"tides-validator","version":"1.0.0","hash":"sha256:..."},  
 "policyPacks":[{"name":"policy-oar","version":"1.0.0"}],  
 "outcome":{"status":"PASS","badge":"A-PASS"},  
 "host":{"pod":"validator-abc","node":"n1"}  
}

### 15.7.2 Redaction Middleware (Python)

SENSITIVE = ["patient", "name", "dob", "mrn", "address", "ssn"]  
  
def redact(msg: str) -> str:  
 out = msg  
 for k in SENSITIVE:  
 out = out.replace(k, "[REDACTED]")  
 return out

### 15.7.3 Immutable Logs

* Append‑only object store with **WORM** retention (≥ 6 years clinical); daily hash chains anchored to a transparency log (optional).

## 15.8 Privacy Compliance (GDPR/HIPAA quick‑map)

| Topic | GDPR | HIPAA | TIDeS Position |
| --- | --- | --- | --- |
| Legal basis | Art. 6/9 | N/A (US) | site policy; typically treatment/research consent |
| De‑identification | Recital 26 | §164.514(b)(2) | documented de‑ID recipes (15.6) |
| Data minimization | Art. 5(1)(c) | minimum necessary | URNs only; linkage KVS separated |
| DPIA | Art. 35 | risk analysis | LINDDUN worksheet + risk register (13.11) |
| Subject rights | Art. 15–22 | access/amend | fulfilled at FHIR/EHR layer, not TIDeS artifacts |

**Normative:** Sites **MUST** document their legal basis and DPIA where GDPR applies.

## 15.9 Data Lifecycle & Retention

1. **Ingest:** DICOM/FHIR enters secure zone; de‑ID applied immediately.
2. **Process:** TIDeS pipeline produces PHI‑free artifacts; validator/report run offline.
3. **Report:** Clinical/Research PDFs generated; PHI‑free; stored in secure archive; optionally re‑attached to EHR via FHIR.
4. **Retention:** Rule packs, fixtures, and reports kept per policy; linkage keys kept separately with stricter controls.
5. **Destruction:** Cryptographic erasure; verify deletion with object store lifecycle logs.

**Lifecycle policy YAML**

retention:  
 bundles: { years: 10 }  
 reports: { years: 10 }  
 audit: { years: 6 }  
 linkage: { years: 15, store: "vault" }

## 15.10 Supply‑Chain Security

* **Reproducible builds:** pinned hashes; requirements.txt with hashes; npm shrinkwrap.
* **SBOMs:** generate for Docker and app; scan with Trivy/Grype.
* **Signatures:** cosign/sigstore on images and release tarballs; policy to verify before deploy.
* **SLSA level:** target SLSA 2+; CI produces provenance (builder digest, inputs, timestamps).

**Cosign policy (notation‑rule)**

attestations:  
 - artifact: ghcr.io/tides/validator:1.0.0  
 require: cosign.signature && cosign.provenance

## 15.11 Backup, DR, and Business Continuity

* **Backups:** encrypted, daily; tested monthly restores; immutable snapshots.
* **RPO/RTO:** site‑defined; validator/reporters re‑deployable from Docker images in < 1 hour.
* **Runbooks:** loss of linkage KVS triggers emergency read‑only mode; reports still viewable; no re‑linking permitted.

## 15.12 Vulnerability & Patch Management

* Weekly scans; triage SLAs: CRITICAL 7d, HIGH 14d, MED 30d.
* Auto‑roll minor patch releases (validator/reporters) with changelog and SBOM updates.
* Security advisories published in SECURITY.md.

## 15.13 Incident Response (IR) Playbooks

**IR‑01 PHI Exposure Suspected** 1. Contain: revoke tokens, block egress, snapshot systems.  
2. Assess: scope, data classes, time window.  
3. Notify: privacy officer, legal; regulatory notifications per jurisdiction.  
4. Eradicate: rotate keys, patch, remove offending configs.  
5. Recover: restore from clean backups; audit post‑mortem; update SOPs.

**IR‑02 Supply‑Chain Compromise** - Quarantine affected images; revert to last signed good; compare SBOMs; force dependency pinning.

**Artifacts:** IR forms and timelines stored in secure evidence vault; hash‑stamped.

## 15.14 Consent, Governance & Data Sharing

* **Research:** store consent metadata (study ID, scope, expiry) separate from TIDeS artifacts; validator can check consent.status when configured.
* **Data use agreements (DUA):** attach DUA ID to exports; block egress if DUA missing.

**Consent check (pseudo‑rule)**

{"id":"consent-required","severity":"ERROR","when":"$.study","test":"{\"==\":[{\"var\":\"consent.status\"},\"granted\"]}","profiles":["B"]}

## 15.15 Privacy‑Preserving Analytics (Research)

* **Aggregation:** publish only aggregate metrics (no voxel arrays) for external sharing unless consented.
* **k‑Anonymity:** ensure k≥5 for small cohorts; suppress small‑cell counts.
* **Differential privacy (optional):** add calibrated noise to summary tables for public releases.

**DP Laplace (Python)**

import numpy as np  
  
def dp\_count(n, eps=1.0):  
 noise = np.random.laplace(0, 1/eps)  
 return int(max(0, round(n + noise)))

## 15.16 Configuration Hardening

* All services read config from **readonly** mounted files; no mutable in‑container secrets.
* Secrets from Vault or KMS; never in env vars in production.
* Disable plugin auto‑loading unless whitelisted; validator --plugins explicit.

**Vault policy (HCL)**

path "tides/\*" { capabilities = ["read", "list"] }  
path "tides/linkage" { capabilities = ["create", "update", "read"] }

## 15.17 Acceptance & Audit Checklist (TIDeS‑SEC‑15)

* PHI‑free artifacts verified (spot check 10 bundles).
* Network egress blocked on validator/reporters.
* KMS keys rotated < 12 months; envelope encryption enabled.
* De‑ID recipes documented and tested on synthetic/real samples.
* Audit logs immutable and centrally stored; sampling alerts if missing.
* SBOMs generated; signatures verified before deploy.
* IR runbooks exercised in tabletop within 6 months.
* DPIA/RA signed (if GDPR).
* Consents/DUAs recorded for research exports.

## 15.18 Frequently Asked Questions

**Q: Can we include a hospital MRN in the DoseReport?**  
A: **No.** Keep MRN in the linkage KVS; inject a display ID at render time only if policy permits.

**Q: How do we let a regulator reproduce results without PHI?**  
A: Provide PHI‑free bundles + signed validator reports + rule pack + fixture corpus; regulator runs validator offline.

**Q: What if our FHIR server requires patient identifiers?**  
A: Use pseudonymous Patient with URN and maintain linkage internally; never propagate direct identifiers into TIDeS artifacts.

## 15.19 Chapter Summary

* TIDeS security & privacy are **built‑in**, not bolted on: PHI‑minimal artifacts, strict network isolation, strong crypto, reproducible de‑ID, signed/audited runs, and clear incident/consent governance.
* These controls are executable via provided configs and code, enabling sites and vendors to **prove** compliance while maintaining scientific and clinical rigor.

**End of Chapter 15 (Normative, Operational & Executable).**

# TIDeS Handbook — Chapter 16

## Change Control & Balloting (RFCs, SemVer, Governance)

**Purpose.** Establish a complete, **executable** change‑management system for TIDeS—including RFC authoring, review and consensus, formal ballots, SemVer rules for text/schemas/rules, errata handling, emergency fixes, and archival with DOIs. This chapter includes normative policy, process timelines, templates, automation, and reference code for maintaining a stable standard while enabling innovation.

**Audience.** Maintainers, Working Group (WG) editors, contributors, vendors, site representatives, and auditors.

**Outcome.** You will: (a) propose and shepherd changes via the RFC process, (b) run ballots and compute results deterministically, (c) apply SemVer versioning across artifacts, (d) publish releases with signed artifacts and DOIs, and (e) preserve traceability across spec, schemas, rule packs, fixtures, and reports.

**Normative keywords:** **MUST**, **SHOULD**, **MAY**.

## 16.0 Governance Model (Normative)

1. **Working Groups (WGs).** Core domains: *Data & Schemas*, *Validator & Rules*, *Interoperability (DICOM/FHIR/OMOP)*, *Security & Privacy*, *Operations & Adoption*. Each WG has **Editors** (≤3) and **Contributors**.
2. **Steering Committee (SC).** 5–9 elected members; holds tie‑break power, sets roadmaps, appoints Editors.
3. **Open Process.** RFCs, ballots, and artifacts are public; meetings are minuted and archived.
4. **COI/Recusal.** Members disclose conflicts; affected members **MUST** recuse from binding votes where directly conflicted.
5. **Decision Records.** Every accepted change yields an **ADR** (Architecture Decision Record) with rationale and links.

## 16.1 Artifact Taxonomy & Versioning (SemVer Normative)

**Artifacts covered** and how SemVer applies:

| Artifact | Version Field | SemVer Bump Rules |
| --- | --- | --- |
| **Spec text** (/spec) | specVersion | editorial = patch; clarifying non‑normative = patch; new SHOULD/MAY = minor; new MUST or breaking = major |
| **JSON Schemas** (/schemas) | $id with version | backward‑compatible keyword additions (optional fields) = minor; required fields/constraints = major; fixes = patch |
| **Validator rule packs** (/validator/rules) | version | new WARN = minor; new ERROR/BLOCK or rule behavior change = major; message copy edits = patch |
| **Profiles** (/profiles/\*) | version | new MAY/SHOULD = minor; new MUST = major |
| **Fixtures** (/examples) | fixturesVersion | new cases = minor; golden output change = **match rule pack** version bump |
| **Reporter templates** (/reporter) | version | cosmetic = patch; structure/content fields = minor; breaking data deps = major |

**Normative:** **Rule IDs are immutable.** A rule’s semantics may only strengthen with a **major** bump; weakening requires deprecation path (see §16.6).

## 16.2 Lifecycle Overview

1. **Idea → Issue.** Open a GitHub issue with the *Proposal* template.
2. **Draft RFC.** Fork /rfcs/RFC-YYYY-NN-title/ with spec diff, schema deltas, rule deltas, migration plan.
3. **WG Review.** Editors triage, request changes, run preview validator.
4. **Public Comment (≥14 days).** RFC published; comment period open.
5. **Ballot.** Eligible voters cast votes; quorum checked; tally produced (§16.4).
6. **Merge/Release.** Upon PASS, Editors merge to main, bump versions as per matrix, regenerate goldens, tag release, mint DOI.
7. **Post‑merge ADR.** Record Final decision with links to ballots, artifacts, and CI runs.

## 16.3 RFC Authoring (Templates & Normative Requirements)

### 16.3.1 Folder Structure

rfcs/  
 RFC-2025-01-dose-rate-flash/  
 README.md # narrative  
 spec-diff.md # redlines (Chapter/sections)  
 schema/ # JSON Schema prototypes  
 rules/expanded-diff.json # new/changed rules  
 fixtures/ # new or updated cases  
 migration.md # guidance, compatibility & timelines  
 impact.md # conformance impact, risk, benefits  
 ballot.yaml # prepared ballot config

### 16.3.2 RFC Markdown Template (README.md)

# RFC-2025-01: Dose-rate (FLASH) Policy Generalization  
  
\*\*Status:\*\* Draft   
\*\*WG:\*\* Validator & Rules   
\*\*Authors:\*\* A. Author (@handle), B. Author (@handle)   
\*\*Discussion:\*\* https://github.com/tides/tides/discussions/1234  
  
## Summary  
Concise description of the change and why it’s needed.  
  
## Motivation & Goals  
Clinical/research value; problems with current spec; target users.  
  
## Design  
- Spec changes (normative text)  
- Schema diffs (new fields, constraints)  
- Rules (new IDs; severities; profile impact)  
  
## Compatibility  
Backward compatibility analysis; migration strategy; deprecation plan if needed.  
  
## Security & Privacy  
Implications and mitigations.  
  
## Alternatives Considered  
Rejected options with rationale.  
  
## Rollout Plan  
Timeline/versions; connectathon tests; site actions.  
  
## Appendix  
References, prototypes, benchmarks.

### 16.3.3 Schema Diff Conventions (Normative)

* Provide **jsonpatch** (RFC‑6902) and **jsonschema‑diff** outputs.
* Mark **required** additions and enum tightening explicitly.
* Include **examples** and validation snippets.

## 16.4 Balloting (Process, Quorum, Tally)

### 16.4.1 Eligibility & Classes

* **Voting classes:** *Implementer* (vendors/sites with running pipelines), *Contributor* (≥3 merged PRs), *Clinician/Physicist* (appointed by SC).
* **Weights:** Implementer=2, Clinician/Physicist=2, Contributor=1.
* **Quorum:** ≥ 60% of active eligible voters across classes; at least 2 Implementers and 2 Clinician/Physicists.

### 16.4.2 Ballot Options

* **YES**, **YES‑WITH‑COMMENTS**, **ABSTAIN**, **NO**.
* Comments are mandatory for **NO**; Editors must respond before close.

### 16.4.3 Deterministic Tally (Normative)

* Score = Σ(weight × vote\_value) with mapping: YES=+1, YES‑WITH‑COMMENTS=+1, ABSTAIN=0, NO=−1.
* **Pass criteria:** Score ≥ 0 and **NO** rate < 20% by headcount; if tie, SC Chair decides.

### 16.4.4 Ballot Record Schema (ballot.schema.json)

{  
 "$id": "https://tides.org/schemas/1.0.0/ballot.schema.json",  
 "$schema": "https://json-schema.org/draft/2020-12/schema",  
 "title": "TIDeS Ballot Record",  
 "type": "object",  
 "required": ["rfcId","open","close","voters","results"],  
 "properties": {  
 "rfcId": {"type":"string"},  
 "open": {"type":"string","format":"date-time"},  
 "close": {"type":"string","format":"date-time"},  
 "voters": {"type":"array","items": {"type":"object","required":["name","class","weight"],"properties":{"name":{"type":"string"},"class":{"enum":["Implementer","Contributor","Clinician"]},"weight":{"type":"number"}}}},  
 "votes": {"type":"array","items": {"type":"object","required":["name","choice"],"properties":{"name":{"type":"string"},"choice":{"enum":["YES","YES-WITH-COMMENTS","ABSTAIN","NO"]},"comment":{"type":"string"}}}},  
 "results": {"type":"object","properties":{"score":{"type":"number"},"pass":{"type":"boolean"},"yesPct":{"type":"number"},"noPct":{"type":"number"}}}  
 }  
}

### 16.4.5 Reference Tally Tool (Python CLI)

#!/usr/bin/env python3  
import json, sys  
MAP = {"YES":1, "YES-WITH-COMMENTS":1, "ABSTAIN":0, "NO":-1}  
  
def tally(ballot):  
 weights = {v['name']: v['weight'] for v in ballot['voters']}  
 head = len(ballot['votes'])  
 no\_count = sum(1 for v in ballot['votes'] if v['choice']=='NO')  
 score = sum(weights.get(v['name'],1) \* MAP[v['choice']] for v in ballot['votes'])  
 ballot['results'] = {  
 'score': score,  
 'pass': score >= 0 and (no\_count/head if head else 1) < 0.20,  
 'yesPct': 100.0\*sum(1 for v in ballot['votes'] if v['choice'].startswith('YES'))/head if head else 0.0,  
 'noPct': 100.0\*no\_count/head if head else 0.0  
 }  
 return ballot  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 b = json.load(open(sys.argv[1]))  
 json.dump(tally(b), sys.stdout, indent=2)

### 16.4.6 GitHub Workflow (Ballot Automation)

name: Ballot Tally  
on: [workflow\_dispatch]  
jobs:  
 tally:  
 runs-on: ubuntu-latest  
 steps:  
 - uses: actions/checkout@v4  
 - run: python rfcs/RFC-2025-01-dose-rate-flash/ballot/tally.py rfcs/RFC-2025-01-dose-rate-flash/ballot.json > rfcs/RFC-2025-01-dose-rate-flash/ballot.result.json  
 - uses: stefanzweifel/git-auto-commit-action@v5  
 with: { commit\_message: "chore(ballot): update tally" }

## 16.5 Editorial vs Substantive Changes (Normative)

* **Editorial:** grammar, links, examples, non‑normative clarifications. **Patch** only; no ballot required.
* **Substantive:** any change to MUST/SHOULD/MAY; schema constraints; rule severities; profile matrices. **Ballot required** with WG review.
* **Emergency Fixes:** security/privacy/clinical safety critical; expedited path (§16.9).

## 16.6 Deprecation, Sunset & Compatibility Windows

1. **Deprecation Notice:** mark rules/fields as deprecated: true (schemas and rule packs) with rationale; severity lowered to INFO where applicable.
2. **Sunset Window:** ≥ 1 minor release before removal; for clinical profile A, **≥ 6 months** real time.
3. **Removal:** major release only; fixtures updated; migration guide provided.
4. **Aliases:** allow read‑only aliases for renamed fields for 1 minor.

## 16.7 Traceability & Change Maps

* **Trace matrix** per change: spec clause ↔ schema pointer ↔ rule IDs ↔ fixtures ↔ reports.
* **Machine‑readable** trace.json committed with each RFC; validator HTML links to corresponding anchors.

**trace.schema.json (excerpt)**

{"type":"object","properties":{"spec":{"type":"array"},"schemas":{"type":"array"},"rules":{"type":"array"},"fixtures":{"type":"array"}}}

## 16.8 Release Orchestration

* **Version bump** PR created by bot based on merged RFC labels (semver:patch|minor|major).
* **Golden regeneration** (Chapter 11) and **report rendering** (Chapter 12) run as pre‑release checks.
* **Tag** vX.Y.Z → GitHub Release → Zenodo DOI mint (Chapter 14).
* **Signed artifacts:** rule packs, schema tarballs, Docker images (cosign).

**Release Bot (pseudo)**

on: [workflow\_dispatch]  
jobs:  
 bump:  
 steps:  
 - uses: actions/checkout@v4  
 - run: python tools/semver\_bump.py --from CHANGELOG.md --labels $(gh pr list --label semver:\* --json labels)

## 16.9 Emergency Changes (Security/Safety)

* **Trigger:** CVE, PHI risk, clinical safety issue (e.g., misinterpretation of units).
* **Process:** SC approves hot‑fix; WG Editors prepare minimal change; **48‑hour** comment window; immediate release X.Y.Z+hotfix.N.
* **Post‑hoc Ballot:** within 30 days to ratify; if failed, revert and publish guidance.

**Hotfix Branching**

main ──┬── v1.0.1  
 └── hotfix/2025-09-25-unit-ucum ──► v1.0.0+hotfix.1

## 16.10 Appeals & Tie‑Breaks

* **Appeal window:** 14 days after ballot result publication.
* **Grounds:** process irregularity, material new evidence, or conflict‑of‑interest violation.
* **Decision:** SC reviews; 2/3 majority to uphold appeal; otherwise ballot stands.

## 16.11 Labels, Milestones & Project Boards

**Labels (authoritative set):** rfc, semver:major, semver:minor, semver:patch, ballot:open, ballot:closed, editorial, security, breaking, interop, schema, rules, docs, fixtures, profiles, blocked, needs-decision.

**Projects:** *Next Minor*, *Next Major*, *Backlog*, *Hotfix*. Cards include RFC ID and links.

## 16.12 Contributor & IPR Policy

* **License:** Code Apache‑2.0; Text/spec CC‑BY‑4.0 (Chapter 0).
* **DCO** (Developer Certificate of Origin) required (sign‑off).
* **Patents:** Contributors grant patent rights per Apache‑2.0 §3 for implementations of contributions.
* **No CLA** required unless organization policy demands; if used, publish CLA terms and automation.

## 16.13 Editorial Workflows & Style

* **Style guide:** imperative rules, consistent UCUM codes, glossary alignment.
* **Spellcheck & linkcheck** in CI; vale or similar linters for prose.
* **Redline previews** generated from RFC spec-diff.md via md-diff job.

## 16.14 Tooling & Reference Code

### 16.14.1 RFC Scaffolder (tools/rfc\_new.py)

#!/usr/bin/env python3  
import sys, pathlib, datetime, re  
slug = sys.argv[1]  
idx = sys.argv[2]  
root = pathlib.Path('rfcs')/f'RFC-{datetime.datetime.utcnow().year}-{idx}-{slug}'  
root.mkdir(parents=True, exist\_ok=False)  
(root/'README.md').write\_text('# RFC: '+slug+'\n\n\*\*Status:\*\* Draft\n')  
(root/'spec-diff.md').write\_text('<!-- Describe redlines by chapter/section -->\n')  
(root/'migration.md').write\_text('## Migration\n')  
(root/'impact.md').write\_text('## Impact\n')  
print('Created', root)

### 16.14.2 SemVer Guard (tools/semver\_guard.py)

import sys, json  
# Read proposed changes metadata and compute required bump  
meta = json.load(open(sys.argv[1]))  
bumps = {'patch':0,'minor':0,'major':0}  
for c in meta['changes']:  
 bumps[c['semver']] += 1  
need = 'major' if bumps['major'] else ('minor' if bumps['minor'] else 'patch')  
print(need)

### 16.14.3 Rule Pack Lint (tools/rules\_lint.py)

import json, sys  
r = json.load(open(sys.argv[1]))  
ids = [x['id'] for x in r['rules']]  
assert len(ids)==len(set(ids)), 'Duplicate rule IDs'  
for x in r['rules']:  
 assert x['severity'] in ['INFO','WARN','ERROR','BLOCK']  
 assert x['id'].islower() or '-' in x['id']  
print('OK')

### 16.14.4 ADR Template

# ADR-2025-03: <title>  
  
## Context  
## Decision  
## Consequences  
## Links  
- RFC  
- Ballot result  
- Release/tag

## 16.15 Transparency & Archives

* **Artifacts:** store ballots, ADRs, RFCs, minutes, and recordings under /governance/archives/YYYY/MM/.
* **Index:** machine‑readable index.json listing object hashes and links.
* **DOI:** tag major/minor releases and upload to Zenodo with CITATION.cff updates.

## 16.16 Communications & Change Notices

* **Release notes** (Chapter 14) plus **Change Notices** summarizing impact for implementers (A4 one‑pager).
* **Deprecation advisories** with timelines and migration steps.

**Change Notice ( template)**

# Change Notice CN-2025-07 — Schema tightening for DoseRate  
\*\*Impact:\*\* Minor   
\*\*Action:\*\* Update exporters to emit `Gy/s` for dose‑rate; run validator v1.1.0   
\*\*Timeline:\*\* enforce ERROR in v2.0.0

## 16.17 Acceptance Checklist — TIDeS‑GOV‑16

* RFC folder complete with schema/rules/fixtures/migration/impact.
* Public comment open ≥ 14 days.
* Quorum met; ballot recorded and tallied; ADR published.
* SemVer bump applied across affected artifacts.
* Goldens and reports regenerated; CI green.
* Release tagged; DOI minted; Change Notice sent.
* Traceability (trace.json) updated and cross‑linked.

## 16.18 FAQs

**Q: Do all rule message copy edits require a release?**  
A: Yes—patch (no ballot) if severity/logic unchanged.

**Q: Can a site ship a site‑specific policy as normative?**  
A: Site packs are allowed but **cannot** downgrade core rules. To standardize, submit an RFC.

**Q: What if two RFCs conflict?**  
A: Editors coordinate; prefer the RFC with broader impact; or merge and ballot as a single combined change.

## 16.19 Chapter Summary

* A rigorous, transparent, and automated change‑control program keeps TIDeS stable yet evolvable. RFCs, structured ballots with deterministic tallying, strict SemVer across artifacts, deprecation windows, emergency hotfix paths, and archival/traceability enable trustworthy, auditable progress for clinical and research ecosystems.

**End of Chapter 16 (Normative & Executable).**

# TIDeS Handbook — Chapter 17

## Glossary, Abbreviations & Controlled Terminology (Authoritative)

**Purpose.** Provide a precise, normative glossary for all terms, symbols, abbreviations, units, and identifiers used in TIDeS. Includes cross‑references to chapters, mappings to external terminologies (UCUM, SNOMED CT, RadLex, LOINC), machine‑readable vocabulary files, and linters to ensure consistent usage across artifacts (schemas, validator messages, reports, and documentation).

**Audience.** Implementers, editors, clinical physicists, NM physicians, QA/regulatory teams, and vendors.

**Outcome.** Consistent, computable language across the specification and software with no ambiguity.

**Normative keywords:** **MUST**, **SHOULD**, **MAY**.

## 17.0 Using This Glossary (Normative)

1. **Canonical forms.** Each entry lists a **Preferred Term** and **Abbreviation** (if applicable). Use the preferred form in normative text and user‑visible messages unless a profile mandates otherwise.
2. **Coding binds.** Where applicable, entries include **coding recommendations** (UCUM, SNOMED CT, RadLex, LOINC). These **MUST** be used in artifacts that carry codes.
3. **Cross‑references.** Each entry may cite relevant chapters/sections.
4. **Machine‑readable vocabulary.** The full set is provided in /spec/vocab/tides-terms.json and /spec/vocab/tides-valuesets.json.
5. **Linting.** The terminology linter in §17.6 **MUST** run in CI; violations are WARN (docs) or ERROR (schemas/rules).

## 17.1 Radiation & Dosimetry Terms

**Absorbed dose** — *Preferred:* Absorbed dose; *Symbol:* D; *Unit:* **Gy** (UCUM Gy). Energy imparted per unit mass.  
**Dose‑rate** — *Symbol:* (); *Unit:* **Gy/s** (UCUM Gy/s). Instantaneous rate of dose delivery (§4).  
**Activity** — *Symbol:* A(t); *Unit:* **Bq** (UCUM Bq). Disintegrations per second.  
**Time‑integrated activity (TIA)** — *Unit:* **Bq·s** (UCUM Bq.s). ( = \_0^{} A(t) , dt ).  
**S‑value** — Mean absorbed dose to a target per unit cumulated activity in a source region; depends on nuclide, geometry, and medium (§5).  
**FLASH** — Ultra‑high dose‑rate regime; default policy threshold (,) for (,) (§4.5, §6).  
**Voxel dose map** — Per‑voxel absorbed dose values on a rectilinear grid (DICOM RTDOSE); UCUM Gy with DoseUnits=GY (§3).  
**DVH (Dose–Volume Histogram)** — Cumulative distribution of dose over structure volume (§12.9).

## 17.2 Imaging & Geometry

**Frame of Reference (FoR)** — DICOM FrameOfReferenceUID; anchors all geometry; **MUST** be preserved across resampling (§3).  
**Spatial Registration** — DICOM objects (rigid/affine/deformable) capturing transforms between image spaces (§3.4).  
**SEG (Segmentation IOD)** — DICOM labelmap encoding ROI masks with coded meanings and algorithm provenance (§3.2).  
**RTDOSE** — DICOM IOD encoding voxel dose values and grid metadata (§3.3).  
**Parametric map** — Image where pixels represent derived parameters (SUV, Ki, TIA) with Real‑World Value Mapping (§3.5).

## 17.3 Kinetics & Modeling

**Mono‑exponential model (monoexp)** — (A(t) = A\_0 e^{-t}); tail integral analytic; fit via WLS (§5).  
**Bi‑exponential model (biexp)** — (A(t) = a e^{-\_1 t} + b e^{-\_2 t}); fit via NLS; TIA from analytic form (§5).  
**Two‑compartment model (2C)** — Compartmental model with exchange rates; optional in Profile A/B (§5.2).  
**Spline / non‑parametric** — Time‑activity curve fitted with smooth basis; integral via quadrature (§5.2).  
**Partial volume correction (PVC)** — Correction for finite spatial resolution; method and coefficients documented (§5.6).

## 17.4 Statistics & Uncertainty

**Type A uncertainty** — Statistical (repeatability, noise) estimated from data (§1.6).  
**Type B uncertainty** — Systematic (calibration, modeling) based on prior knowledge (§1.6).  
**Delta method** — Propagation of uncertainty through differentiable functions; provides approximate variance (§5.8).  
**Bootstrap** — Resampling technique to estimate confidence intervals for model parameters or dose metrics (§5.8).  
**AIC (Akaike Information Criterion)** — Model selection metric; lower is better (§5.7).  
**R² (Coefficient of Determination)** — Fit goodness indicator (§5.7).

## 17.5 Interoperability & Data Exchange

**UCUM** — Unified Code for Units of Measure; **MUST** be used for all quantities (§1.1, §7.2).  
**SNOMED CT** — Clinical terminology for anatomy/conditions (ROI coding).  
**RadLex** — Radiology lexicon—sometimes preferred for imaging concepts.  
**LOINC** — Lab/observation codes; use 89469-4 for Absorbed dose (§7.2).  
**FHIR** — HL7 Fast Healthcare Interoperability Resources; use DiagnosticReport, Observation, ImagingStudy (§7.2).  
**OMOP CDM** — Observational Medical Outcomes Partnership Common Data Model; tides extension tables (§7.3).

## 17.6 Identifiers, URNs & Hashes

**URN** — urn:tides:<type>:<UUID> where <type> ∈ {subject, study, imaging, seg, dosemap, dosereport, pkmodel, provenance} (§2).  
**Bundle hash** — SHA‑256 digest of bundle.json, encoded as sha256-<hex>; used in badges and reports (§10, §12).  
**Rule ID** — Lowercase hyphenated identifier for validator rules (e.g., unit-ucum, registration-fuid); **immutable** (§9, §16).

**Regex (Normative):**  
- URN: ^urn:tides:[a-z]+:[0-9a-fA-F-]{8,}$  
- UCUM Gy: ^Gy$  
- UCUM Gy/s: ^Gy/s$

## 17.7 Safety & Policy Terms

**OAR (Organ at Risk)** — Structures with dose constraints (kidney, liver, marrow) with versioned policy limits (§6).  
**Constraint‑WARN/ERROR** — Validator decisions when near or exceeding limits (§6, §9).  
**Pediatric modifiers** — Age/weight adjustments applied by policy pack (§6).  
**Cumulative tracking** — Longitudinal accumulation across studies (§6).

## 17.8 Validator & Conformance Language

**MUST / SHOULD / MAY** — RFC 2119/8174 normative keywords (§0, §10).  
**Profile A/B/C** — Conformance profiles: A=Clinical‑Full, B=Research‑Voxel, C=Legacy‑Organ (§10).  
**Badge** — Signed evidence of conformance (e.g., A-PASS) (§10).  
**Finding** — Individual rule evaluation outcome with severity ∈ {INFO,WARN,ERROR,BLOCK} (§9).  
**Waiver** — Pre‑approved exception that downgrades a matched ERROR/BLOCK for badge calculation (§10.6).

## 17.9 DICOM Essentials (Tag Quick Reference)

| Concept | Tag | Keyword | Notes |
| --- | --- | --- | --- |
| Frame of Reference UID | (0020,0052) | FrameOfReferenceUID | **MUST** exist for dose maps |
| Dose Units | (3004,0002) | DoseUnits | GY for absolute dose |
| Grid Frame Offset Vector | (3004,000C) | GridFrameOffsetVector | Required for RTDOSE grid spacing |
| Dose Grid Scaling | (3004,000E) | DoseGridScaling | Scale factor for stored values |
| Segmented Property | (0062,0003) | SegmentedPropertyCategoryCodeSequence | ROI coding |

**Normative:** For Profile A/B, RTDOSE and SEG instances **MUST** contain the listed tags where applicable (§7.1).

## 17.10 FHIR Essentials (Profile Pointers)

| Resource | Purpose | Profile URI (example) |
| --- | --- | --- |
| DiagnosticReport | Theranostics summary report | http://tides.org/fhir/StructureDefinition/tides-diagnosticreport-theranostics |
| Observation | Absorbed dose value | http://tides.org/fhir/StructureDefinition/tides-absorbed-dose |
| ImagingStudy | DICOM references | http://tides.org/fhir/StructureDefinition/tides-imagingstudy |

**Unit binding:** UCUM Gy, Gy/s (§7.2). **Code binding:** LOINC 89469-4 for absorbed dose.

## 17.11 OMOP Essentials (TIDeS Extension)

**Tables:** tides\_study, tides\_dose, tides\_roi\_dose (§7.3).  
**Keys:** study\_id (text), person\_id (int ↔ site‑specific pseudonym), roi\_code (text).  
**Constraints:** UCUM units implied; values stored in SI; uncertainty stored as percent.

## 17.12 Units & Prefixes (UCUM Quick Map)

| Quantity | SI Unit | UCUM Code | Notes |
| --- | --- | --- | --- |
| Absorbed Dose | gray | Gy | 1 Gy = 1 J/kg |
| Dose‑rate | gray per second | Gy/s | Gy.s-1 equivalent accepted |
| Activity | becquerel | Bq | 1 Bq = 1 s⁻¹ |
| Time | second | s | ISO‑8601 for timestamps |
| Length | millimetre | mm | spacing/reporting |

**Normative:** **No implicit prefixes**; emit explicitly (e.g., mGy only if schema allows and rule pack recognizes unit). Default TIDeS uses **Gy** for dose (§1.1).

## 17.13 Abbreviations & Symbols

AIC — Akaike Information Criterion (§5.7).  
AUC — Area Under Curve (synonym of TIA when context is activity; prefer TIA).  
CI — Confidence Interval (§5.8).  
CIS — Clinical Information System (contextual).  
DICOM — Digital Imaging and Communications in Medicine (§3, §7).  
DP — Differential Privacy (§15.15).  
DVH — Dose–Volume Histogram (§12.9).  
FoR — Frame of Reference (§3).  
FHIR — Fast Healthcare Interoperability Resources (§7).  
HU — Hounsfield Unit (§5.5).  
JSONPath — JSON query language used in rules (§9).  
MC — Monte Carlo (§5.4).  
NLS — Nonlinear Least Squares (§5.2).  
OMOP — Observational Medical Outcomes Partnership (§7.3).  
OPA — Open Policy Agent (§13, §15).  
PK — Pharmacokinetics (§5).  
PVC — Partial Volume Correction (§5.6).  
ROI — Region of Interest (§3.2).  
RTDOSE — Radiotherapy Dose object (§3.3).  
SEG — Segmentation IOD (§3.2).  
TIA — Time‑Integrated Activity (§1, §5).  
UCUM — Unified Code for Units of Measure (§1.1).  
URN — Uniform Resource Name (§2).  
VNA — Vendor Neutral Archive (PACS).

## 17.14 Controlled Valuesets (Machine‑Readable)

**/spec/vocab/tides-valuesets.json (excerpt)**

{  
 "$schema":"https://json-schema.org/draft/2020-12/schema",  
 "$id":"https://tides.org/spec/vocab/tides-valuesets.json",  
 "version":"1.0.0",  
 "valuesets": {  
 "unit-dose": {"system":"http://unitsofmeasure.org","allowed":["Gy"]},  
 "unit-dose-rate": {"system":"http://unitsofmeasure.org","allowed":["Gy/s","Gy.s-1"]},  
 "roi-codes": {  
 "system":"http://snomed.info/sct",  
 "allowed":[{"code":"64033007","display":"kidney"},{"code":"74281007","display":"liver"}]  
 }  
 }  
}

**Normative:** Validators and exporters **MUST** reference this file (by version) or a site‑approved superset; narrowing is allowed only via policy packs.

## 17.15 Terminology Linter (Executable)

**Purpose:** Catch non‑canonical spellings/units and missing codes in docs and JSON artifacts.

**tools/term\_lint.py**

#!/usr/bin/env python3  
import sys, re, json, pathlib  
VOC = json.loads((pathlib.Path('spec/vocab/tides-valuesets.json')).read\_text())  
CANON = {  
 'dose': 'Absorbed dose', 'doserate': 'Dose-rate', 'frameofreference': 'Frame of Reference',  
 'flash': 'FLASH', 'uncertainty': 'Uncertainty', 'dvH': 'DVH', 'rtDose': 'RTDOSE', 'seg': 'SEG'  
}  
BAD\_UNITS = re.compile(r'\b(Gray|GY|gy|Gy/s^)|mGy\b')  
  
errors = 0  
for path in sys.argv[1:]:  
 txt = pathlib.Path(path).read\_text(errors='ignore')  
 if BAD\_UNITS.search(txt):  
 print(f"ERROR[{path}]: Non-UCUM or disallowed unit variant detected"); errors += 1  
 for k,v in CANON.items():  
 if re.search(rf"\b{k}\b", txt, flags=re.I):  
 # optionally suggest canonical form  
 pass  
sys.exit(1 if errors else 0)

**CI Hook (pre‑commit)**

- repo: local  
 hooks:  
 - id: term-lint  
 name: term-lint  
 entry: python tools/term\_lint.py spec/\*\*/\*.md  
 language: system  
 pass\_filenames: true

## 17.16 Style & Notation Conventions

* **Quantities**: use roman symbols (A, D, ()); vectors bold; scalars italic in math.
* **Timestamps**: ISO‑8601 Zulu (UTC); show timezone if local policy differs.
* **Decimals**: period decimal separator; thousands separator optional; 1 decimal place for organ Dmean/Dmax unless more is justified.
* **Identifiers**: always lowercase URNs; UUID v4 recommended.

## 17.17 Deprecated & Legacy Terms

* **AUC (for activity)** — *Deprecated wording*; use **TIA** for clarity.
* **mGy for dose** — *Deprecated in TIDeS artifacts*; internal computations may use mGy but outputs **MUST** be Gy (§1.1).
* **Classic PET IODs** — *Discouraged*; prefer Enhanced IODs (§3.1).

## 17.18 Editorial Guidance for Authors

* Prefer **short, active** definitions; avoid synonyms unless listed here.
* When introducing a term, include code bindings if relevant (UCUM/SNOMED/LOINC).
* Link to this glossary entry on first use in a chapter.

## 17.19 Acceptance Checklist — TIDeS‑GLOSS‑17

* All units are UCUM and appear in approved valuesets.
* All ROIs use SNOMED/RadLex codes with human‑readable labels.
* All validator rules referenced by **ID**; no ad‑hoc paraphrases.
* URNs conform to regex; hashes recorded where required.
* Linter passes in CI with zero **ERROR**.

## 17.20 Chapter Summary

This glossary provides the **single source of truth** for TIDeS terms and codes. It ensures precise, computable language across artifacts and implementations, backed by machine‑readable valuesets and CI linting to prevent drift.

**End of Chapter 17 (Normative & Executable).**

# TIDeS Handbook — Chapter 18

## Acceptance Criteria & “10/10 Executable” Readiness

**Purpose.** Define the final, **objective** bar for calling a TIDeS Implementation Guide “10/10 Executable.” This chapter provides normative acceptance criteria, evidence bundle formats, automated readiness checks, certification workflows, connectathon tests, and reproducibility gates. It ships with scripts and schemas so sites and vendors can self‑attest, and auditors can verify byte‑for‑byte.

**Audience.** Site/vender leads, QA, auditors, certifying bodies, working‑group editors.

**Outcome.** A deterministic, repeatable process to declare and prove readiness for Profiles A/B/C with signed artifacts.

**Normative keywords:** **MUST**, **SHOULD**, **MAY**.

## 18.0 Readiness Definition (Normative)

A deployment (site or product) is **10/10 Executable** when **all** of the following are true:

1. **Spec Completeness.** Chapters 0–17 implemented with no TODOs; schemas match spec text; rule pack IDs and severities synchronized (Ch.8–9).
2. **Validator Strength.** All MUST rules for the target profile(s) enforced; WARNs implemented where defined; BLOCK reserved for catastrophic safety/PHI violations (Ch.9–10,15).
3. **Fixtures & Goldens.** All 10 canonical cases (Ch.11) ship, generate, and validate with byte‑exact golden outputs (HTML/JSON/TXT/SARIF) under CI.
4. **Reporting.** Clinical (2‑page) and Research (≤4‑page) reports render deterministically from the same JSON inputs, pass accessibility checks, and embed validator summaries (Ch.12).
5. **Security/Privacy.** PHI‑free artifacts, zero‑egress validator/reporters, signed audit logs, envelope encryption available, de‑ID recipes documented/tested (Ch.15).
6. **Interop.** DICOM, FHIR, OMOP mappings operational with at least one round‑trip test each; FHIR resources validate against TIDeS profiles (Ch.7).
7. **Governance.** SemVer matrix applied; change control active; CITATION + DOI attached to the release (Ch.0,14,16).
8. **Docs & DX.** Repo layout canonical (Ch.14), docs site builds with no errors, contributor/security policies present.
9. **Operations.** SOPs, monitoring, KPIs and DR plans adopted; acceptance tests pass (Ch.13).
10. **Glossary & Lint.** Terminology linter clean; valuesets bound; UCUM only (Ch.17).

## 18.1 Evidence Bundle (Authoritative Format)

**Purpose:** Portable proof of readiness.

**Layout**

.evidence/  
 manifest.json # machine‑readable index (schema below)  
 validator/  
 case\_\*/report.json  
 case\_\*/report.txt  
 case\_\*/report.html  
 case\_\*/report.sarif  
 fixtures/  
 case\_\*/bundle.json  
 reports/  
 case\_\*/clinical.pdf  
 case\_\*/research.pdf  
 security/  
 sbom/\*.spdx.json  
 policies/\*.yaml # de‑ID, network, retention  
 audits/\*.json # immutable run logs  
 signatures/  
 \*.sig # Ed25519 signatures for artifacts  
 release/  
 CITATION.cff  
 LICENSE  
 CHANGELOG.md

**manifest.schema.json**

{  
 "$id":"https://tides.org/schemas/1.0.0/evidence-manifest.schema.json",  
 "$schema":"https://json-schema.org/draft/2020-12/schema",  
 "type":"object",  
 "required":["version","profile","artifacts"],  
 "properties":{  
 "version":{"type":"string"},  
 "profile":{"type":"string","enum":["A","B","C"]},  
 "specVersion":{"type":"string"},  
 "rulePackVersion":{"type":"string"},  
 "artifacts":{"type":"array","items":{  
 "type":"object","required":["rel","sha256"],  
 "properties":{"rel":{"type":"string"},"sha256":{"type":"string"},"sig":{"type":"string"}}  
 }}  
 }  
}

## 18.2 Readiness CLI (Executable)

### 18.2.1 tools/readiness\_check.py

#!/usr/bin/env python3  
import json, subprocess, sys, hashlib, os  
from pathlib import Path  
  
ROOT = Path(\_\_file\_\_).resolve().parents[1]  
EV = ROOT/'.evidence'  
MAND\_CASES = [  
 'case\_pass\_minimal','case\_fail\_units','case\_registration\_ok','case\_registration\_link\_missing',  
 'case\_sampling\_inadequate','case\_safety\_violation\_kidney','case\_provenance\_missing',  
 'case\_ucum\_wrong\_unit\_text','case\_flash\_flag\_coverage','case\_legacy\_profile\_C']  
  
FAIL=0  
  
# 1) Validate fixtures and compare against goldens  
for c in MAND\_CASES:  
 bundle = ROOT/'examples'/c/'bundle.json'  
 if not bundle.exists():  
 print(f'[X] Missing bundle for {c}'); FAIL+=1; continue  
 # run validator  
 profile = 'A' if c != 'case\_legacy\_profile\_C' else 'C'  
 res = subprocess.run([sys.executable,'-m','validator.cli','validate',str(bundle),'--profile',profile,'--format','json'], capture\_output=True, text=True)  
 if res.returncode not in (0,2):  
 print(f'[X] Validator non‑pass for {c}: exit {res.returncode}'); FAIL+=1  
 # golden compare (JSON)  
 got = json.loads(res.stdout)  
 want\_p = ROOT/'.golden'/c/'report.json'  
 if want\_p.exists():  
 want = json.loads(want\_p.read\_text())  
 got.get('provenance',{}).pop('issuedAt', None)  
 want.get('provenance',{}).pop('issuedAt', None)  
 if got != want:  
 print(f'[Δ] Golden mismatch for {c}'); FAIL+=1  
 # write into evidence  
 out\_dir = EV/'validator'/c; out\_dir.mkdir(parents=True, exist\_ok=True)  
 (out\_dir/'report.json').write\_text(res.stdout)  
  
# 2) Render clinical & research reports deterministically  
rend = subprocess.run([sys.executable,'tools/render\_all.py'], capture\_output=True, text=True)  
if rend.returncode!=0:  
 print('[X] Report rendering failed'); print(rend.stdout); print(rend.stderr); FAIL+=1  
  
# 3) Security posture quick checks  
sec\_ok = True  
np = ROOT/'deploy/k8s/tides-core.yaml'  
if np.exists() and 'deny-egress' not in np.read\_text():  
 print('[!] Missing deny-egress policy'); sec\_ok=False  
FAIL += 0 if sec\_ok else 1  
  
# 4) Manifest build  
artifacts=[]  
for p in EV.rglob('\*'):  
 if p.is\_file():  
 artifacts.append({ 'rel': str(p.relative\_to(EV)), 'sha256': hashlib.sha256(p.read\_bytes()).hexdigest() })  
mani = {'version':'1.0.0','profile':'A','specVersion':'1.0.0','rulePackVersion':'1.0.0','artifacts':artifacts}  
(EV/'manifest.json').write\_text(json.dumps(mani, indent=2))  
  
print('\nReadiness:', 'PASS' if FAIL==0 else f'FAIL ({FAIL})')  
sys.exit(FAIL)

### 18.2.2 GitHub Workflow Gate

name: Readiness Gate  
on: [workflow\_dispatch]  
jobs:  
 gate:  
 runs-on: ubuntu-latest  
 steps:  
 - uses: actions/checkout@v4  
 - uses: actions/setup-python@v5  
 with: { python-version: '3.11' }  
 - run: pip install -r validator/requirements.txt jinja2 weasyprint  
 - run: python tools/readiness\_check.py

## 18.3 Certification Workflow (Badge Issuance)

1. **Submit** an Evidence Bundle (.evidence/) with signed manifest.json to the Certifying Body (CB).
2. **Automated Verification:** CB re‑runs the **Readiness CLI** against submitted fixtures; reproduces validator results and report hashes.
3. **Security Review:** CB checks de‑ID policy, deny‑egress config, audit samples.
4. **Decision:** CB issues signed badge JSON per target profiles.

**Badge JSON** (immutable; also in Chapter 10)

{  
 "profile":"A",  
 "status":"A-PASS",  
 "specVersion":"1.0.0",  
 "rulePackVersion":"1.0.0",  
 "issuedAt":"2025-09-25T13:10Z",  
 "subject":"site:Acme-Medical",  
 "evidenceHash":"sha256-…",  
 "signature":"ed25519:…"  
}

**Verification Tool** (tools/badge\_verify.py)

import json, sys  
from nacl.signing import VerifyKey  
  
badge = json.load(open(sys.argv[1]))  
key = bytes.fromhex(open('certs/cb-pubkey.hex').read().strip())  
msg = ('|'.join([badge[k] for k in ['profile','status','specVersion','rulePackVersion','issuedAt','subject','evidenceHash']])).encode()  
VerifyKey(key).verify(msg, bytes.fromhex(badge['signature']))  
print('OK')

## 18.4 Connectathon Test Plan

**Goals:** Prove real‑world interop across DICOM/FHIR/OMOP, validator parity, and reporting.

**Tracks & Criteria**

* **DICOM Track** — Export Enhanced NM/PET with SEG + RTDOSE; cross‑vendor FoR alignment; **Criteria:** round‑trip FoR equality; SEG label code parity.
* **FHIR Track** — POST DiagnosticReport & Absorbed Dose Observations; **Criteria:** profile validation success; UCUM only.
* **OMOP Track** — ETL tides\_dose rows; **Criteria:** row‑count parity and value equivalence within 1e‑6.
* **Validator Track** — Run 10 fixtures; **Criteria:** byte‑for‑byte golden matches.
* **Report Track** — Render PDFs; **Criteria:** hash equality, PDF/UA tag validity.

**Scoring:** Each track worth 2 points; total ≥ 8 points required for a Connectathon PASS.

## 18.5 Reproducibility & Determinism

* **Seeds fixed** for any stochastic components (bootstraps); record seeds in provenance.
* **No net access** during validation and rendering; CI asserts offline mode.
* **Byte equality** for HTML/PDF except timestamp fields explicitly excluded.

**Provenance addition**

{"seeds":{"bootstrap":1234,"plot":2025}}

## 18.6 Accessibility & Localization Checks

* HTML reports score ≥ 90 in Lighthouse Accessibility; CLI prints score.
* PDFs validate with veraPDF to PDF/UA where possible.
* Language packs load with no missing keys; fallback coverage 100%.

**tools/a11y\_check.sh**

#!/usr/bin/env bash  
set -e  
npx -y lighthouse out/report.html --only-categories=accessibility --quiet --output=json --output-path=out/lh.json  
jq '.categories.accessibility.score' out/lh.json

## 18.7 Security/Privacy Readiness (Quick Audit)

**Automated checks:** - Detect PHI patterns in bundles/reports/logs (regex heuristic + allowlist).  
- Verify Kubernetes NetworkPolicies **deny egress**.  
- Verify encryption policies and rotation periods in config.  
- Presence of IR runbooks and audit trail samples.

**tools/sec\_quick\_audit.py**

import re, sys, json, pathlib  
PHI = re.compile(r"(MRN|DOB|PatientName|SSN|Address)")  
problems = 0  
for p in pathlib.Path('examples').rglob('\*.json'):  
 if PHI.search(p.read\_text(errors='ignore')):  
 print('[X] Potential PHI in', p); problems += 1  
print('SEC PASS' if problems==0 else f'SEC FAIL ({problems})')  
sys.exit(1 if problems else 0)

## 18.8 Publishing & DOI Minting

* **Release tag** created (vX.Y.Z).
* **Artifacts** uploaded (schemas, validator tarball, Docker image digests, evidence template).
* **Zenodo deposition** created with metadata from CITATION.cff.
* **DOI** written back into CITATION.cff and README badges (Ch.14).

## 18.9 Final Acceptance Checklist — TIDeS‑ACC‑18

**Spec/Schema/Rules** - [ ] specVersion synchronized across spec, schemas, validator rules, profiles.  
- [ ] Rule IDs immutable; severities per Chapter 9; profile matrices per Chapter 10.  
- [ ] Schemas validate all canonical bundles; negative fixtures fail as expected.

**Fixtures/Reports** - [ ] 10 fixtures generate correctly; goldens matched; DVHs render (if configured).  
- [ ] Clinical & Research reports rendered; PDF/UA checks pass; localization coverage ≥ 1 language besides English.

**Security/Privacy** - [ ] PHI‑free confirmations; deny‑egress policies in place; audit logs signed.  
- [ ] De‑ID recipes executed on sample DICOM & FHIR.

**Interop** - [ ] DICOM SEG/RTDOSE round‑trips;  
- [ ] FHIR DiagnosticReport/Observation POST and profile‑valid;  
- [ ] OMOP ETL aligns to DDL and row counts verified.

**Governance/Release** - [ ] CHANGELOG updated; SemVer applied; DOI minted; badges generated.  
- [ ] ADR recorded; traceability map updated.

**Operations** - [ ] SOPs live; KPIs tracked; DR runbook tested.  
- [ ] Readiness Gate workflow green; Evidence Bundle produced and signed.

## 18.10 Common Pitfalls & Remedies

| Pitfall | Symptom | Remedy |
| --- | --- | --- |
| Golden drift | CI diffs on validator HTML/JSON | Re‑pin rule pack version; regenerate goldens only with SemVer bump |
| Non‑deterministic PDFs | Hash mismatch across runs | Freeze fonts, page numbers, images; remove timestamps from DOM |
| PHI leakage | Regex finds MRN/DOB | Fix de‑ID pipeline; add redaction middleware; re‑audit |
| FoR mismatch | Registration rule ERROR | Confirm spatial registration; align RTDOSE FoR with imaging |

## 18.11 Sunset & Maintenance

* Readiness must be **re‑verified** at every **minor** or **major** release.
* Evidence Bundles expire after **12 months** or upon any policy/rule pack change—whichever is sooner.
* Certification body may spot‑check sites with synthetic replays.

## 18.12 Chapter Summary

This chapter sets the enforceable bar for “10/10 Executable.” With the evidence bundle, readiness CLI, security/interop checks, connectathon plan, and certification process, any stakeholder can **prove** conformance, reproducibility, and safety without PHI exposure.

**End of Chapter 18 (Normative & Executable).**